NeXT

Systems Support

June, 1990

NeXT Systems Support

Table of Contents

Introduction **1**

Class Objectives 1
Course Outline 2

Overview of User Configurations **7**

User Configurations 7

Physically Setting up a NeXT Computer System **8**

Product Description 8
Options and Accessories 10
Product Specifications 11
Care and Feeding of the NeXT System 14

Using the NeXT Interface **15**

Using the NeXT Interface 15
Keyboard Overview 29
Less than One Hundred and One Things You Can Do with a Mouse 31
Less than One Hundred and One Things You Can Do to a Dock 33
Less than One Hundred and One Things You Can Do to a Window 34
Standard Window Components 36
Less than One Hundred and One Things You Can Do to a Menu 37
Standard Menu Components 39
Less than One Hundred and One Things You Can Do to an Application 40
Relationships Between Window and Application Functionality 41
The Preferences Application 42
Edit 51
Hints for Using a NeXT Computer 58

Disk Administration **59**

Mounting File Systems 59
File System Components 61
Structure of a Directory File 62
Three Parts of a File 63

Becoming a Power User **64**

NeXT Directory Structure 64
Shell Tubes 65
Accessing the UNIX Shell on a NeXT Computer 66

Basic Security **68**

File Permissions 68
su man page 72

Process Monitoring and Control **73**

Image of a Process 73
Life Cycle of a Process 74
Process Monitoring and Control 75
User id and Process id 80
Login and Logout Processes 81
What's in /etc/ttys 82
Determining Permissions 84

System Startup and Shutdown **85**

ROM Monitor Commands 85
Monitor Command Sequence 88
Monitor Map 89
Booting from a Local Disk 90
Startup Scripts 92
Daemon Processes 93

Configuring a Multi-User Workstation **94**

User Information 94
Group Information 95
Housekeeping Files 96

Overview of NeXT Functionality and Features **98**

The Problems with Current Desk Top Software 98
Shell Sequence of Interpretation 99
Modular Components of the UNIX Operating Environment 100
Mach and UNIX 101
System Overview 102
NeXT Software 103
Maximizing Software Efficiency 104
NeXT CPU Board Layout 105

Configuring a NetInfo Network **106**

Client and Server Process Interaction 106
Man page - ping 107
Man page - telnet 108

User Administration Using NetInfoManager **112**

The NetInfo Suite of Manager Applications 112

Advanced Disk Administration **113**

Monitoring Disk Usage 113
Man page - df 115
Man page - du 116
Ejecting an Optical Disk 117
Overall View of Memory 118
Man page - fsck 119

Configuring NetBoot Clusters **121**

Booting from a Network 121

Troubleshooting **123**

On-Screen Troubleshooting 123
Shutdown Procedures 124
Tracking System Error Messages 125
Man page - lpc 126
Shorting NVRAM in Case of Forgotten Hardware Password 128
UNIX Command Summary 130

Serial Communications **132**

Special Files for Serial Connections 132
Serial Interface Information Flow 134
Cabling Diagram 135
Using a Modem with the NeXT Computer 136
Man page - tip 139

Interacting with NeXT **142**

Support Programs 142

Lab Materials **143**

Using Bundled Applications 143
Using Mathematica 151
Introduction to Interface Builder 156
Man page - arp 166
Man page - netstat 167
Man page - FTP 169

NeXT Systems Support
Class Objectives

At the conclusion of this course all attendees will be able to:

- Use the NeXT computing environment.
- Describe and demonstrate the unique benefits and features of the NeXT Computer System.
- Administer a stand-alone NeXT Computer system.
- Administer a network of NeXT Computer systems.
- Function as a fully qualified NeXT support person.

NeXT Systems Support
Course Outline

Introduction
- find bathrooms, lunch, dinner, and factory tour
- identify class participants
- ask and list questions

Overview of User Configurations
- describe available user configurations

Physically Setting up a NeXT Computer System
- understand the license
- care for and feed your NeXT Computer System
- put the pieces together (including printer)
- turn the power on and off
- describe the basic power on sequence

Using the NeXT Interface
- identify and move the various screen objects
- use the mouse
- move, close and miniaturize windows
- move, populate and depopulate the dock
- use menus
- launch, hide and quit an application
- traverse the directory structure with the Browser
- use the Workspace Manager to copy, move and delete files
- identify and use the various special keys
- use Edit to perform basic editing
- use the standard Print Panel to print and select printing options
- send text, voice, graphics and file attachments with Mail
- search on-line documentation with the Digital Librarian
- use the Workspace Manager to kill applications
- use the Preferences application to change system options
- develop a personal style for working on a NeXT Computer

Disk Administration

- describe the purpose and functionality of mountable filesystems
- use the Workspace Manager to mount and eject an optical disk
- use the Workspace Manager to copy files between disks
- use the Workspace Manager to create and restore from backups
- use BuildDisk to build a bootable disk

Becoming a Power User

- use Workspace Manager to set the view to display all files
- use Preferences to become a UNIX expert
- use the Browser to locate system files and directories
- use Shell and Terminal to issue UNIX commands

Basic Security

- use Preferences to add a password for the me account
- define users, groups, and permissions
- describe the special purpose of the root and me accounts
- use the Workspace Manager to work with file and directory permissions
- login and switch user to root at the appropriate times
- describe the function of the wheel group

Process Monitoring and Control

- define a process
- list and describe the possible states of an application
- use UNIX commands to list and kill processes
- describe the lineage of a given process
- describe the relationships between loginwindow, windowserver, workspace manager, and other user applications
- restart the WindowServer and loginwindow
- describe how permissions are determined

System Startup and Shutdown

- use the ROM monitor to set the hardware password
- use the ROM monitor to set security parameters
- use the monitors to manually boot the system
- observe and troubleshoot system start-up processes and scripts
- describe the purpose of the various daemon processes

Configuring a Multi-User Workstation

- list the components of a user account
- use UserManager to add, delete and modify user accounts
- appropriately modify the contents of the ~/.NeXT directory
- modify the default new user set up with the directory /usr/template/user

Overview of NeXT Functionality and Features

- describe the NeXT design goals
- describe NeXT's software features
- describe NeXT's hardware features
- describe NeXT's networking features
- describe the benefits of "starting from scratch"

Configuring a NetInfo Network

- describe the utility of sharing data about machines over the network
- use NetManager to set up a NeXT configuration server
- use NetManager to set up automatic host addition
- use NetManager to set up a clone configuration server
- use Mail to send and receive remote mail
- use PrintManager to install a network printer
- use the Print panel and Printer application to choose a printer
- use UNIX commands to debug a troubled network

User Administration using NetInfoManager

- match tasks with the appropriate Manager application
- use NetInfoManager to view user data
- use NetInfoManager to add a user to and delete a user from an existing group
- use NetInfoManager to remove a password
- use NetInfoManager to create a new group

Advanced Disk Administration

- describe the purpose and use of device files
- use UNIX commands to mount and unmount file systems
- implement a working backup strategy
- use UNIX commands to check and repair file systems
- use UNIX commands to determine file system usage

Exporting and Importing Directories with the Network File System

- describe the functionality of network file systems
- describe possible uses of a network file system
- set up an /etc/exports file to export directories
- set up NetInfo to mount (import) remote directories
- use UNIX commands to monitor and control remote mounting
- describe NFS security and limitations

Configuring Network-Wide User Accounts

- describe the utility of sharing data about users over the network
- set up an /etc/exports file to export users' home directories
- set up NetInfo to mount users' remote home directories
- use UserManager to add a network user
- use UserManager to modify existing accounts to become network accounts
- use MailManager to prepare a computer to be a mail server
- use MailManager to set up mail clients
- use NetInfoManager to create system-wide mail aliases

Configuring NetBoot Clusters

- describe the net boot process
- set up a net boot server
- boot a system off the network

Troubleshooting

- troubleshoot hung applications
- use UNIX commands and nmi monitor to shutdown an uncooperative system
- find and use error messages
- use UNIX commands and ROM monitor to eject an uncooperative MOD
- use UNIX commands to troubleshoot the printer
- eliminate a hardware password using physical means

Serial Communications

- modify the /etc/ttys file to support a serial terminal
- modify the /etc/ttys file to support an incoming modem
- use tip to connect to a remote system over a serial line
- use tip to transfer files to and from a remote system over a serial line
- use PrintManager to support a serial printer
- describe the functionality of the UUCP package

Connecting a NeXT Computer to a Heterogeneous IP Network

- connect to a non-NetInfo network
- configure a NeXT as a Yellow Pages client
- configure a NeXT as a client of the Internet Domain Name Service
- configuring a NeXT in a non-NetInfo, non-yp, non-BIND network
- access a remote printer on a non-NeXT computer

Interacting with NeXT, Inc.

- get technical support from NeXT
- report bugs/suggestions to NeXT

Optional Labs

Using Bundled Software

- perform basic editing with WriteNow
- define words using Digital Webster
- index files with the Digital Librarian
- evaluate equations with Mathematica
- produce documents with FrameMaker
- produce a simple demonstration application with Interface Builder

Using the Network

- use ftp to transfer files to and from a remote computer
- use UNIX command lines to launch a NeXT application connected to a remote window server
- explore advanced networking topic of your choice

User Configurations

Personal Workstation - The NeXT Computer can be used "as shipped" for stand-alone development without providing for individual file ownership and protections.

Shared Workstation - The Next Computer can be used as a shared development environment by multiple users. Just use Preferences to add a password to the "me" account and use the UserManager application (which can be found in the /NextAdmin directory) to add users. See the printed *Network and System Administration* document for more information on setting up a Shared Workstation.

Networked Workstations - NeXT Computers can be networked together and/or with other computers running TCP software so that users can do remote access, remote file transfers, remote mail and remote printing (remote mail and printing are currently limited to homogeneous networks). To accomplish this, networked NeXT Computers need to share information. This is accomplished most easily with NeXT's custom network database called NetInfo or, less easily, with SUN's Yellow Pages. Use NetManager's Local Configuration panel to configure each NeXT as a NetInfo Configuration Server, NetInfo Client, or Yellow Pages Client. NetInfo Clients can be added just by attaching the new host to a network with an existing configuration server, booting the new host, and following the on-line instructions. See the printed *Network and System Administration* document for more information on setting up Networked Workstations.

Network Users - A NeXT Computer can be part of either a homogeneous or heterogeneous network that supports networked users. Such a network must provide a mechanism for sharing file systems as well as network administration data that describes, amongst other things, the users of the network. We use NFS to share file systems and NetInfo or Yellow pages to share administrative data. See the printed *Network and System Administration* document for more information on setting up networked users.

Product Description

The NeXT Computer combines the best attributes of personal computers and workstations, adds features previously found only on mainframe computers, and introduces entirely new innovations.

The NeXT Computer System begins with the Computer, a one-foot cube that houses the main CPU (central processing unit) board (with room for three additional expansion boards of the same size); a universal power supply, which makes the system usable throughout the world; and up to two 5.25-inch full-height mass storage devices. Standard in the cube are eight megabytes (MB) of main memory, expandable to 16 MB on the main CPU board.

The computer is powered by Motorola s top-of-the-line microprocessor and memory management unit, the 68030, and the 68882 Floating-Point Unit for fast mathematical computation, both running at 25 megahertz (MHz). The system also includes a 10 MIPS (million instructions per second) Motorola 56001 Digital Signal Processor, which supports complex, computation-intensive processes including CD-quality music and sound synthesis. Also standard is high-performance Ethernet networking.

The CPU board contains two proprietary VLSI (very large-scale integration) chips that endow the NeXT Computer System with mainframe-like qualities. One, called the Integrated Channel Processor (ICP), ensures high system throughput by carefully managing and optimizing the flow of data within the system, particularly between the main memory, the CPU and peripheral devices such as the network, sound output, monitor and disk drives.

The other VLSI chip, called the Optical Storage Processor (OSP), controls the system s read/write/erasable 256 Megabyte Optical Disk. The Optical Disk represents a new form of mass storage technology, combining laser technology and magnetic (Winchester disk) technology. Information on the Optical Disk can be edited and manipulated, not just read, and the entire disk can be removed and carried between computers for convenience and security.

The NeXT Computer System also features the MegaPixel Display, with a 17-inch, crisp, high-resolution screen. Images on the screen can appear as black, white and various shades of gray, through the use of two bits per pixel. This shading ability adds depth to the images that appear on the MegaPixel Display, which in turn adds depth to an observer s interpretation.

In addition to its visual capabilities, the MegaPixel Display houses impressive sound capabilities. The system can generate CD-quality stereo sound (i.e., 44.1 kHz, 16-bit two-channel digital audio). Users can listen to the sound through a built-in speaker, through stereo headphones via a built-in headphone jack or through a connected audio system via built-in line-out jacks. The Display also features a microphone jack for voice input. Because the MegaPixel Display derives its power from the cube, it requires no separate AC plugs or external power sources.

Taken together, the hardware components of the NeXT Computer System create a powerful, efficient and aesthetic platform that allows users to extend far beyond their traditional desktop computer boundaries.

Options

The 400dpi laser printer

An optional but recommended component of the NeXT Computer System is the 400 dpi Laser Printer. As its name implies, the printer can produce extremely high-quality output of 400 dots per inch (dpi), as well as the standard desktop laser printer output of 300 dpi.

Memory Expansion

Memory on the NeXT Computer is expandable to 16MB using 1MB, 100 nanosecond, page memory, nonparity SIMMs. These are available in packs of 4MB.

Mass storage options

NeXT offers both 330MB and 660MB SCSI hard disks for the NeXT Computer. These high-performance storage devices provide the added storage capacity necessary in server and heavy development environments.

Accessories

Blank media

Optical disk cartridges, which have a 256MB capacity, are available individually, in packs of 10, or packs of 40.

MegaPixel Display

Ordered as a separate item, this accessory includes the MegaPixel Display, keyboard, mouse, and 3-meter display cable.

Ethernet Connector Kit

This kit includes a T-connector, 30 feet (9 meters) of thin ethernet cable, and a terminator.

Product Specifications

Computer

Processors

Motorola 68030 25 MHz CPU
Motorola 68882 25 MHz FPU
Motorola 56001 25 MHz DSP
NeXT integrated channel processor
    12 DMA channels
    32 MB/sec bandwidth
NeXT optical storage processor

Memory

64Kbytes boot ROM
8 to 16 MB of memory, user expandable in 4 MB increments
256K bytes of dual-ported video RAM

Communications and Interfaces

Video monitor interface (DB19)
Thin wire Ethernet ® (BNC), IEEE 802.3 compatible
Two RS-422 serial ports (SCC chip Z8530) with 8-pin mini-DIN connectors
SCSI interface implemented with a 53C90 SCSI chip with transfer rate of 4.8 MB/sec (burst rate), DB 25 external, 50-pin shrouded vertical header internal
Four enhanced NuBus expansion slots (three after main processor board), type C Eurocard connector operating at a basic cycle rate of 12.5 MHz, a burst rate of 25 MHz and a peak transfer rate of 100 MB/sec.
400 dpi Laser Printer port (DB9) with 5 Mbits/sec serial transfer rate
Digital signal processor port (DB15) with a maximum transfer rate of 2Mbits/sec

Power

Powers up to four slots with 25 watts each
Automatically adjusts to line frequency and voltage
Voltage: 90 to 270 volts AC
Frequency: 47 Hz to 63 Hz single phase
Power: 300 W (including MegaPixel Display)

Clock/calendar
32.768 KHz crystal powered by a 3.0 volt removable lithium battery

Environment
Ambient temperature: 32° to 104 ° F (0 ° to 40 ° C)
Relative humidity: 10% to 90% non-condensing
Altitude: 0 to 15,000 feet (0 to 4,500 meters)

Regulations
UL listed and CSA certified
Complies with FCC Part 15 Class A requirements

Size and weight
Weight: 25 to 40 lbs (10 to 19 Kg) depending on peripheral configuration
Height x width x depth: 12 x 12 x 12 (30 x 30 x 30 cm)
Can hold two full-height, 5.25 storage devices

MegaPixel Display

Monitor
17 " monochrome, flat screen
1120 x 832 pixels, 2-bit resolution (black, dark gray, light gray, white)
94 pixels per inch
100 MHz video bandwidth
68.3 Hz vertical refresh rate

Input/output
Video monitor interface (DB19) carries +12/-12 VDC, video, and I/O
Keyboard jack, 5-pin mini-din connector
16-bit, 44.1 kHz stereo output through gold-plated RCA phono line-out jacks
Mini-connector stereo headphone jack
Integrated speaker
8-bit, 8 kHz audio input via monophonic microphone jack

Keyboard/mouse
85-key low-profile keyboard, including cursor keys, numeric keypad, brightness and volume controls, and power on/off
Two-button opto-mechanical mouse

Size and weight
Weight: 50 lbs (23 Kg)
Height x width x depth: 17.3 x 16.0 x 14.0 (44 x 40 x 35 cm)

Mass Storage

Magneto-Optical Disk Drive
256 MB (formatted)
92 ms average seek time
5 ms average seek time within 5 MB range
1.14 MB/sec raw burst transfer rate
0.26 – 0.83 MB/sec raw sustained transfer rate
3000 RPM
Infinite read/write/erase
Removable, primary storage and/or backup device

SCSI Hard Disks
Rotary voice coil actuator and integrated SCSI contoller for speed and reliablity
Formatted capacities: 330 MB and 660 MB
Average seek time: 14.5 and 16.5 ms respectively
5 and 8 platters respectively
45 KB dual ported FIFO buffer
4.8 MB/sec raw burst transfer rate
3600 RPM

Care and Feeding of the NeXT System

The NeXT Computer is designed to be rugged and reliable, but some care should be taken to avoid scratching surfaces. The Cube is coated with a water-based black paint which can be scratched with sharp objects. Do *not* use solvents such as alcohol, ammonia, or any abrasives to clean the computer. Repeated use of these substances will cause the paint to soften and rub off. We recommend water-based soap cleaners such as Formula 409.

The computer will operate best if not exposed to extreme temperatures. Keep it out of direct sunlight. You should not stack the cube on top of the NeXT LaserPrinter, or put the LaserPrinter on top of the cube. Cubes may be stacked atop one another but should not be stacked more than three high. Make sure that nothing blocks the vents on the bottom of the cube, to allow for adequate air flow. If the cube is placed on a shag carpet, it is a good idea to put a piece of cardboard or wood underneath the computer to prevent excessive dust or fibers from getting inside.

Thumb screws on cable connectors should be tightened fully (but not excessively). Be sure to avoid excessive crimping or stressing of any cables. Stressing or crimping the cable between the cube and the monitor can cause distortions in the display.

The NeXT Computer is designed to withstand normal static conditions. However, if service work is being performed with the back of the cube removed, full static-safe procedures should be followed.

Do *not* defeat the purpose of the grounded power cord. The three-prong plug should only be connected to grounded electrical outlets.

Never transport a cube with an optical disk inserted in the optical disk drive. Serious damage can be done to the drive mechanisim.

See *The NeXT User's Reference Manual* for further information about keeping the NeXT Computer in good condition.

Using the NeXT Interface

Introduction

As you sit facing a NeXT computer, many questions are probably crowding your mind. The first might even be, "What are all those *things* on my screen?". Moving to more specific issues, you may be wondering how to rearrange your screen layout, access files or run applications. Over and over, the phrase "How do I ..." is undoubtedly repeating itself in your brain. Never fear—the answers to these and many other questions are at hand. Once you have completed this module, you will be able to perform basic operations on the NeXT computer: identify (and move) screen objects, access files and applications and copy, move, rename and delete files. So, limber up those fingers and prepare for your introduction to the NeXT interface.

Introducing the Workspace Manager

The items you see before you on your screen are brought to you courtesy of the Workspace Manager. The Workspace Manager is an application program that greets you when you access a NeXT computer, and remains active as long as you are using the system. The function of the Workspace Manager is to provide access to all of the files and applications stored on your system. A similar service is provided by the shell in a traditional UNIX environment, and by Finder in the Macintosh environment.

- **Identifying the objects:** At this point you should see four objects of interest displayed on your screen:

  Menu — In the upper left corner is a *menu* associated with the Workspace Manager, which allows you to issue commands (note that it is labeled **Workspace**).

  Window — Near the center of your screen is a *directory window*, labeled **Directory Browser**, which provides a view into the directory structure.

  Dock — On the right edge of the screen is a column of icons, called the *dock*, which provides easy access to commonly used applications.

  Black Hole — Finally, a special icon called the *black hole* is provided for file removal (as the bottom icon of the dock).

- **Using the mouse:** Now that you've examined your screen, it's time to begin exploring. The first thing to do is select an item from the main menu. Use the mouse to position the cursor over the menu item labeled **Info ....** Next, press and release either mouse button (this is called *single clicking*, or just *clicking*). As a result of your action, a *panel* has appeared on the screen. On this panel is a button in the upper right corner marked with an X. Use the mouse to single click this button. The panel disappears, and your screen is once again in its original configuration. (The button you clicked is a *close button*—more on it later.)

- **Moving the dock:** All of the objects on your screen can be moved (within limits). Start testing the extent of these limits by relocating the dock:

  Use the mouse to position the cursor over the NeXT icon
  Press down on a mouse button
  Move the mouse until the dock is in a desirable location
  Release the mouse button

  This process of pressing the mouse button and moving the mouse is called *dragging*. Specifically, the four steps of dragging are:

  1) Position the cursor
  2) Press a mouse button
  3) Move the mouse
  4) Release the button

  As you have just seen, moving an object involves dragging. Try moving the dock a few more times. You have undoubtedly discovered two fascinating things (at least) about the dock:

  The dock can only be moved up or down, and the NeXT logo cannot be moved off the screen.

  This icon is a permanent fixture of the dock. Moving any of the other icons may cause them to disappear. (If this happens to you, don't worry—you'll see how to replace them a little later.)

- **Moving windows:** The next item to throw around is the directory window. Across the top of this window is a black band that contains two buttons and a title. This band is called the *title bar*. Moving a window is achieved with the same method that you used with the dock, with one constraint:

  When moving a window, the cursor must be placed over the title bar before you begin dragging.

  Do not, however, click either of the buttons contained in the title bar (one, you will remember, closes the window, the other will be explored a little later). How far can you drag the directory window? Can you remove it from the screen?

Try moving the menu now, using the same method (again, dragging on the title bar). Can the menu be pulled off the screen?

As you may have noticed, the menu, directory window and panel all look pretty similar. Each is a special kind of *window*. Like all windows, they are each rectangular and have a title bar.

Just about any visual object can be dragged partially off the screen, as long as some part of the object is always visible. For those items with a title bar, part of the bar must remain visible.

Using the Browser

- **Examining the directory structure:** As mentioned earlier, the directory window is a view into the directory structure. With this facility, you can locate and access the files and applications on your system. The directory structure used on the NeXT machine is a hierarchical directory tree based on the UNIX file system.

A directory tree is a method of organizing files that involves a topmost directory pointing at files and other directories (called subdirectories). These subdirectories can themselves point to files or additional subdirectories (which may point to additional subdirectories, and so on). Thus, you have a directory structure containing many different levels, all of which can be traced back to a common starting point.

There are many ways to represent such a structure. Outline format is a possibility, as is a tree with branches and leaves, or nesting folders within folders within folders. The directory window on your screen is currently displaying a standard *Browser view*.

> *The standard Browser representation has the advantage of displaying several different levels of the directory hierarchy simultaneously within a single window.*

In this representation, the leftmost column is topped by the label /. This character is used to represent the *root* directory, the topmost level of the directory hierarchy. This column is a listing of the file and subdirectory names contained in the root directory. The subdirectory name **me** is highlighted. The column to the right of / is labeled **me** and displays the contents of that directory. To the right of the columns is an icon of a folder labeled with the name **me**. This area is called the *Browser icon well*.

- **Traversing the directory tree:** Use the mouse to move the cursor until it is over the word **NextLibrary** in the leftmost column of the Browser view. Notice the arrowhead to the right of the name. Watch the screen carefully as you click once. Several things happen:

  The word **NextLibrary** is highlighted
  The title of the second column of the window changes to **NextLibrary**
  The second column displays the contents of the directory (folder) **NextLibrary**
  The icon well on the right changes to be a folder with the name **NextLibrary**

  Find the word **Documentation** in the **NextLibrary** column, and click that. Again, the selected name is highlighted, the rightmost column displays the contents of the indicated directory, and the icon well shows the selected directory name. The full path of this directory can be read from the column headings across the top of the Browser: **/NextLibrary/Documentation.** As you might expect, this is where the on-line documentation can be found

  In the **Documentation** column, click **NeXT** (again note the arrowhead), and observe what happens. All the columns shift to the left to make room for the new directory.

  In the **NeXT** directory, select **SysRefMan.**

- **Scrolling a column of names:** At the bottom of every column in the Browser is a pair of *scroll buttons*. Place the cursor over the left button at the bottom of the **SysRefMan** column. Note that this button has a black arrowhead pointing down. Now, press down on a mouse button and watch the names scroll by as you hold it down (holding down a mouse button is called *pressing*).

  Pressing an active scroll button produces a continuous scrolling of text.

  In this case, the text is a list of names in a directory. Try scrolling up the list. Notice that the arrowhead on a scroll button turns gray when it is inactive, and black when it is active.

- **Scrolling the Browser horizontally:** On the left edge of the Browser view are two additional scroll buttons. These function very similarly to the other scroll buttons. Experiment with horizontal scrolling by alternately pressing the two buttons.

- **Selecting a file:** Use your traversing skills to move to the directory **/NextApps**—scroll the Browser to the left until the root directory appears, then click the name **NextApps.** Notice that the rightmost column is cleared.

  In this directory, locate the name **Webster.** Notice that there is no arrowhead to the right of this name. Click on **Webster** (make sure that it is only a single click) and observe.

  No new column was created for you this time. Instead, the name was highlighted (as usual), and the icon well changed to display a picture of a terminal. **Webster** is the name of the file containing the Digital Webster application.

  > *Whenever an arrowhead appears to the right of a name in the Browser view, it indicates a subdirectory. Names without this indicator are files.*

  Compare the icon displayed in the Browser icon well to those on your dock. Is there a match?

  Select another file in this directory by single clicking its name. Is the icon for this application stored on your dock?

  Continue selecting file names and comparing icons until you have identified several of the icons populating the dock.

- **Launching an application from the Browser:** Once you have finished your explorations, select the application **Webster** by clicking its name in the directory window. Now, move the cursor over the icon in the Browser icon well and press twice on either mouse button in very rapid succession (this is called *double clicking*). The icon will immediately flash white, indicating that the icon has been double clicked. The icon on the dock will turn into a "ghost" version of itself (a very pale image), indicating that the application is being launched. (If you removed the **Webster** icon from your dock in earlier explorations, the icon will appear somewhere near the bottom of your screen. More about this later.) A short time later, the application will become active, and several

things will happen at once. The **Webster** menu will appear in the top left corner of the screen, a window will appear on some other part of the screen, and the docked icon will return to its normal color. Notice, however, that this icon no longer has three little dimples in its lower left corner.

The presence of three dimples on a docked application icon indicates an inactive application. The absence of these dimples indicates that the application is running.

- **Quitting an application from the menu:** You'll be able to explore this application later. For now, exit the program by selecting the **Quit** command from **Webster**'s main menu (notice the dimple reappearing on the docked icon).

  Almost all applications can be exited by selecting the **Quit** *option from their main menu.*

- **Depopulating the dock:** Locate the icon associated with the **WriteNow** application on your dock (if you removed it earlier in your explorations, just select some other icon). Using the dragging method, begin to pull the icon away from the dock. Notice that a ghost version of the icon remains on the dock as you begin to pull away. Release the mouse button while the ghost image is still on the dock. Watch as the icon "snaps" back onto the dock in its original position.

  Drag the icon again, this time pulling it far enough away so that the ghost image disappears. Now release the mouse button, and the icon itself vanishes. This is called *depopulating* the dock. (Don't worry, you're not really eliminating any files or applications, simply eliminating access to that application from the dock.)

  Removing an icon from the dock is accomplished by dragging the icon away from the dock until the ghost image disappears and then releasing the mouse button.

- **Populating the dock:** Now, place the cursor over the **WriteNow** icon displayed in the Browser window. Using the dragging method, pull the icon toward the dock. Notice that the ghost image of the icon appears in the dock as you approach. As soon as the ghost image appears, you can release the mouse button, and the icon will be placed on the dock.

  Whenever a dragged icon approaches the dock, a ghost image appears in the position where the icon will be placed when the mouse button is released.

  If you have removed any other icons from your dock (and remember what they look like), use this method to restore your dock to its original condition. As it is changed, the contents of the dock are remembered so that it will look the same when next you log in. As you gain more experience with the system, you will undoubtedly determine your own favorite dock configuration.

- **Launching an application from the dock:** Launching an application from the dock is achieved with the same procedure as launching from the Browser icon well. Go ahead and double click the **WriteNow** icon on the dock. Do you notice anything different about the way the application is launched? You shouldn't; the functionality is exactly the same.

- **Quitting an application using the keyboard:** Examine the menu for the **WriteNow** application. Notice that to the right of the **Quit** option is the letter **q**. Exit the application by holding down the **Command** key (lower left or right on the keyboard) and pressing **q**. This is called a *keyboard alternative*.

  Whenever a letter appears to the right of a menu item, it is an indication that that menu item can be selected by pressing the indicated letter in conjunction with the **Command** *key.*

- **Using additional directory windows:** There will be times when one directory window is just not enough—copying or moving files between directories requires multiple directory windows. To open a second directory window, place the cursor over the Browser window (avoiding its buttons) and click. This makes the Browser active (more on this later). Now, single click the word **NextApps** in the leftmost column. The icon well will be changed to display the folder icon for **NextApps**. Place the cursor over this icon and double click. The icon will flash white and a second window will appear. This second window should be partially overlaying the original directory window.

Double clicking a directory icon in the Browser icon well opens an additional directory window.

It is entirely possible that this window does not look anything like the original Browser view (and if it does, it soon won't).

The standard Browser view with which you have become familiar is only one of several available representations. To change the way the directory information is presented, place the cursor over the **View** command in the **Workspace** menu and click. Note that this item has an arrowhead to its right, indicating a submenu. In the submenu are listed the various ways in which the directory information can be shown. Select one of them and observe your new view. One by one, select each of the other available views, comparing them to ones you have already seen. Which is your favorite?

- **Determining the key window:** After you've reviewed all the possible views, make sure the view for your second window is set to **Browser**. You may have noticed that your original window now has a grey title bar, while the new window has a black bar. The black title bar indicates which window on the screen is the *key window*.

  The key window is the window that will receive keystrokes and be affected by menu commands. It is identified by a black title bar.

  Remember, only the key window was affected when you changed the directory representation with the **View** command. Changing the key window is a very simple matter—move the cursor so that it is anywhere within the borders of your original directory window (except over either of its buttons) and click once. The title bar of the original window turns black and the entire window moves to the foreground—it has become the key window. Any keyboard or menu actions taken now will affect this window (go ahead and change the view, just to prove it, but be sure to change the view back to **Browser**). Now, click anywhere within your second window (again, avoiding its buttons) to make it the key window.

- **Closing a window:** Now that you've got a second directory window, how do you get rid of it? Right—click the close button (remember it's the button in the top right corner of each of your windows marked with an X). This process is called *closing* a window, and the button you clicked is a *close button.* Be sure you don't confuse closing a window with quitting an application. Just because you've closed the Browser window doesn't mean that you have quit Workspace Manager.

  Clicking a close button closes a window.

- **Copying a file:** One of the most popular activities on a computer system is to copy files. Copying files on the NeXT system involves opening a second directory window. So, first make sure that your directory window has its view set to **Browser**. Then, make sure that the current directory in that window is **/NextApps**. Now, open a second window by double clicking the icon **NextApps**. In this second window, make sure the view is set to **Browser** and then change the current directory to be your home directory (**/me**) .

  Make the original view the key window by clicking anywhere within its borders. In the directory **NextApps**, once again locate the file **Webster** and single click its name. Notice that the associated icon appears in the icon well.

  At this point, you need to be able to see the icon well in both windows. If this is not the case, move the key window until it is.

  Now, to copy the file **Webster** into your home directory, place the cursor over the **Webster** icon and drag the icon toward the icon well in the second window. As the icon approaches, watch the folder open up to receive the new file as the cursor changes to represent two small sheets of paper. As soon as this occurs, release the mouse button. If the operation takes more than a second or two, a panel will appear, indicating that the system is **Copying** the file **Webster.**

  Copying a file is accomplished by dragging a file's icon from one directory window to another.

Ta-da—you now have your very own copy of the **Webster** application residing in your home directory. Notice that the original **Webster** file was not affected, and the word **Webster** now appears in the listing of your home directory.

- **Moving a file:** But wait—you didn't really want the file to be in your home directory, you wanted it to be in the **Apps** directory *under* your home directory. You now need to move the file from its current location to the **Apps** directory.

  First, make sure your original Browser view is the key window (the one pointing to **NextApps**). Then, make the current directory of that window be **/me/Apps.**

  Make your second window the key window and make sure the file **Webster** is selected in your home directory. To move the file, simply place the cursor over the icon in the well, hold down the **Command** key, press down the mouse button and drag the icon to the folder in the original directory window. Once again, the folder opens to receive the new file (but the cursor does not change). Release the mouse button and the **Command** key (in that order). (You must release the mouse button *before* you release the **Command** key, because the significance of the **Command** key is only noted at the time the mouse button is released.) After you release the button, if the operation is lengthy, a **Moving** panel will appear. This time, the file is moved from one directory to another, so that the file no longer appears in your home directory.

  Moving a file is accomplished by dragging a file's icon from one directory window to another while pressing the **Command** *key.*

- **Renaming a file:** Now that you've got this spiffy new file, how about giving it a spiffy new name? Make the view containing the new file the key window. Make sure the icon for the **Webster** file appears in the icon well, then move the cursor so that it is over the file name under the icon. Double click the name, and it will be highlighted. Now, enter **Dictionary**. Notice that, as you type, an insertion point appears (a blinking vertical bar) that indicates where your characters will be placed.

  A double click selects an entire word, allowing you to overwrite it.

  Press **Return** and an Alert panel will appear informing you that changing the name of this file will cause problems for the Workspace manager and asking you if you want to change the name anyway. Since we are working with a copy of the application, changing the name won't cause any problems, so you can go ahead and select **Yes.** Watch as both

the name and the icon are changed in the Browser. The icon changes because the original icon was associated with an application file named **Webster**. The new icon is a generic one used for files containing applications that do not have their own icon specified.

Use the mouse to place the cursor immediately before the **D** in **Dictionary**. Click once. Now enter **My** followed by **Return**.

A single click positions the insertion point and allows you to insert or delete characters.

- **Deleting a file:** Now that you've gone to all the trouble of copying, moving and renaming this file, you discover that you don't really need it. To get rid of the file **MyDictionary**, place the cursor over the associated icon in the Browser, then drag it towards the black hole. As the icon approaches, watch the hole begin to spin. As soon as this occurs, you can release the mouse button, and the icon will be sucked into the black hole. Once again, a **Moving** panel may appear, indicating that the file is being moved into the black hole.

Deleting a file is accomplished by dragging the associated icon to the black hole.

- **Recovering a file:** Sending a file to the black hole seems pretty final, but is it? Can a file ever be recovered? Place your cursor over the black hole and double click. Surprise—what you see now is another directory window containing the names of all the files that have been deleted. (For those of you who are curious, the black hole is actually a directory to which deleted files are moved. For the insatiably curious, the name of this directory is **~/.NeXT/.NeXTtrash**.) At this point, you can drag a file out to a different directory to recover it. Go ahead and try moving the file **MyDictionary** from the black hole back to your **Apps** directory (by dragging its icon). Then, move it into the black hole once again.

Files that have been deleted can be recovered by double clicking the black hole and then dragging the appropriate icon from the resulting directory window into another directory window.

When you're finished looking at this window, you'll want to turn it back into the black hole. To accomplish this, find the button at the left of the title bar that has a black bar across its top. Click once, and the black hole returns. (This button is the *miniaturize* button. More on it later.)

- **Destroying files:** Well, if removed files still exist, don't they take up space? Certainly. Can you remove them permanently? Of course.

  To destroy files dragged to the black hole, select **Utilities** *from the Workspace Manager menu, and then* **Destroy Deleted Files**.

So, has the file truly been destroyed? How can you find out? You got it—double click the black hole to turn it into a directory window. Once you've verified that the file is really gone, turn the view back into the black hole.

Before moving on, clean up the Workspace a little—close all directory windows but one, and make sure that the directory for that view is **/NextApps**.

- **Using the Browser's text window:** Now it's time to play with a couple of shortcuts for selecting files. Type in an upper case **S**. A new window appears on your screen labeled **Name Expansion; Return Opens.**

The name **Shell** is highlighted in the **NextApps** column, and the icon well now holds the associated icon. This is just the same as if you had single clicked the name **Shell**. Finish typing in the word **`Shell`**, but do NOT press **`Return`**.

Typing in a file name selects that file, just as clicking its name in a directory window does.

Close the **Name Expansion** window by clicking the close button, and then enter an upper case **`L`**. Now, press the **`Esc`** key. The name in the window expands to be **Librarian.app**, and that file is selected in the Browser view.

Close the window once again, and then enter a capital **`W`** followed by the **`Esc`** key. Rather than being expanded into a name, the system beep is sounded, indicating that the expansion was not successful (there are two files that begin with W). Even so, the first file that matches is highlighted (**Webster**).

The **Esc** key is the expansion key. When you enter one or more characters, the first file with a name matching the entered character(s) is selected. If you follow the characters with the **Esc** key, the name is expanded to whatever file matches. If no files have a name matching the characters you've entered, the expansion will not work. The same is true if more than one file matches (you must enter enough characters to identify the file uniquely).

Now, for the grand finale. Close the **Name Expansion** window, type an **S** and press **Return**. The name is expanded to **Shell**, the file is highlighted, and then the application is launched, all in one fell swoop.

> *Entering an application's file name in the Name Expansion window, followed by a Return, selects the file and launches it, just as double clicking its name in a directory window does.*

Conclusion

At this point, you have explored the very basics of dealing with your NeXT system—using the mouse, selecting items, moving objects, manipulating the dock, traversing the directory structure, using menus and windows, and accessing applications. Hopefully, there are several fewer questions rattling around in your brain than when you started. You are, obviously, far from finished, but you now possess the basic tools necessary to begin a thorough exploration of the NeXT computer.

Keyboard Overview

The NeXT keyboard looks much like ordinary typewriter and computer keyboards and is used to enter text, perform software operations that can also be performed with the mouse, and perform hardware operations such as turning the system on and off. The keyboard is divided into three different sections: the regular keypad, the system control keys and the numeric keypad.

Character Keys: Character keys are labeled with letters, numbers, punctuation marks, or other typographical symbols and are used to enter text.

Special Character Keys: Special character keys generate non-printing characters that perform special functions depending on the context in which they are used. These keys include:

Return Pressing this key most often simply starts a new line of text. In other contexts it is used to tell the system to interpret a command line.

Enter Pressing this key often has the same effect as pressing **Return**. It is used by some applications to initiate the interpretation of a command line. **Command-Return** generates the same special character as **Enter**.

Tab Pressing this key moves to the next preset tab stop. **Shift-Tab** moves the cursor in the reverse direction.

Delete This key is most often used to delete previously entered text or other objects. **Shift-Delete** produces a backspace character which usually produces the same effect as **Delete**.

Esc The Escape key is most often used to change modes when you are communicating with a UNIX utility.

Modifier Keys: Modifier keys do not generate characters but change the effect of the mouse or other keys. They are used by holding them down while pressing another key. For example, holding down the **Command** key while pressing the **h** key will hide the current application. This action is indicated with the notational convention **Command-h**. These modifier keys include:

Shift This key is used to produce capital letters and the upper character of keys labeled with two symbols.

Alternate This key is most often used to produce special-purpose, infrequently used characters, such as scientific and mathematical symbols. See Chapter 2 of *The NeXT User's Reference Manual* for coverage of the available symbols.

Command This key is used chiefly to provide a keyboard alternative to making a menu selection with the mouse. Several of the more commonly used keyboard alternatives are labeled in green on the keyboard.

Control This key is most often used when you are communicating with UNIX to perform traditional UNIX functions (e.g., stopping a job with **Control-z**).

Arrow Keys: Arrow keys are most often used to move the insertion point in the indicated direction when you are entering text or graphics.

System Control Keys: The five system control keys modify the computer's physical functions:

Power This key turns the computer on and off. See the handout *Shut Down Procedures* for more information.

Volume These keys adjust the volume of the speaker contained in the MegaPixel display. The upper key increases volume, the lower one decreases it. Holding down the **Command** key and pressing the lower volume key will mute the speaker.

Brightness These keys adjust the brightness of the MegaPixel display. The upper key increases the brightness, the lower one decreases it.

Less Than One Hundred and One Things You Can Do with a Mouse

There are really only two simple actions that can be performed with a mouse:

- Moving the mouse, which results in the cursor moving in an analogous direction
- Pressing and releasing either of the mouse buttons.

These two simple actions can be combined to produce the following four basic mouse-initiated operations:

Clicking: Clicking the primary mouse button results in the selection of an object, while clicking the secondary mouse button results in the appearance of a popup menu at the cursor's location. Clicking is accomplished by using the mouse to move the cursor to the desired location and pressing and releasing the appropriate mouse button. The **Preferences** application is used to determine which mouse button is which. The default is that both buttons act as the primary mouse button. If the popup menu button is enabled in **Preferences**, then the **Handedness** option determines whether the left or right button is the primary button.

Multiple clicking: Multiple clicking is used to extend the meaning of a single click. For example, positioning the cursor on an icon in the workspace and double clicking results in the application associated with that icon being selected and launched.

Dragging: Dragging with the mouse is accomplished by placing the cursor over an object, pressing the primary mouse button, moving the mouse, and then releasing the mouse button. Dragging is used to:

- Move an object. For example, a scroll bar or workspace icon.
- Define a range. For example, a range of text can be selected by dragging from the first to the last character you wish to select.
- Slide from one object to another. For example, you can slide from a menu item to the submenu that appears when the initial item is selected.

Pressing: Pressing an object, like multiple clicking, results in an extension of the function performed by clicking on that object. Pressing is accomplished by positioning the cursor over the desired object and holding the primary mouse button down for some period of time. For example, pressing a scroll button results in continuous scrolling until the mouse button is released.

Less Than One Hundred and One Things You Can Do to a Dock

The NeXT dock is used to provide easy access to frequently used applications. An application's icon can be docked (placed in the dock) and that application can then be launched by double clicking the docked icon. The dock is defined as the area below the NeXT logo on the right side of your screen.

Moving the dock: The dock can be moved up and down the right side of your display. Simply position the cursor over the NeXT logo and drag up or down. Notice that the NeXT logo cannot be moved off the screen. This icon provides a handle that is always available to move the dock back onto the screen.

Docking an application: To dock an application, grab the application's icon from the Browser and position it somewhere near the dock. Notice that a "ghost" version of the icon appears in the dock as your selected icon approaches the dock. This is where the icon will be placed when you release the mouse button. Once an application's icon has been docked, it will remain docked, through logging out and logging in, through sleet and rain and dark of night—until you explicitly remove it from the dock.

Undocking an application: To remove an application's icon from the dock, grab the application's docked icon and drag it to any location not in the dock's region.

Peeking behind the dock: For those of you who care, the names of all docked applications are stored in a file called **~/.NeXT/.dock**. This file is updated whenever you change the dock, so that it will be identical when you next log in. Be warned, this is a non- ASCII file.

Less Than One Hundred and One Things You Can Do to a Window

The following is a list of the most common things users do to windows:

Opening a standard window: In most cases, standard windows are automatically opened when you launch an application. Many applications also provide a menu selection for opening a new window or a window containing previously stored contents.

Closing a window: When you are done working with a window you will want to close it. Closing a window has the effect of removing it from the screen. Almost all applications provide a mechanism for closing their standard windows. In most cases these mechanisms include a close button and a close menu selection. Many panels also include a close button.

Moving a standard window: All windows can be moved anywhere on your screen. Simply drag the window by any portion of its title bar except its buttons (you wouldn't want to push anyone's buttons accidentally).

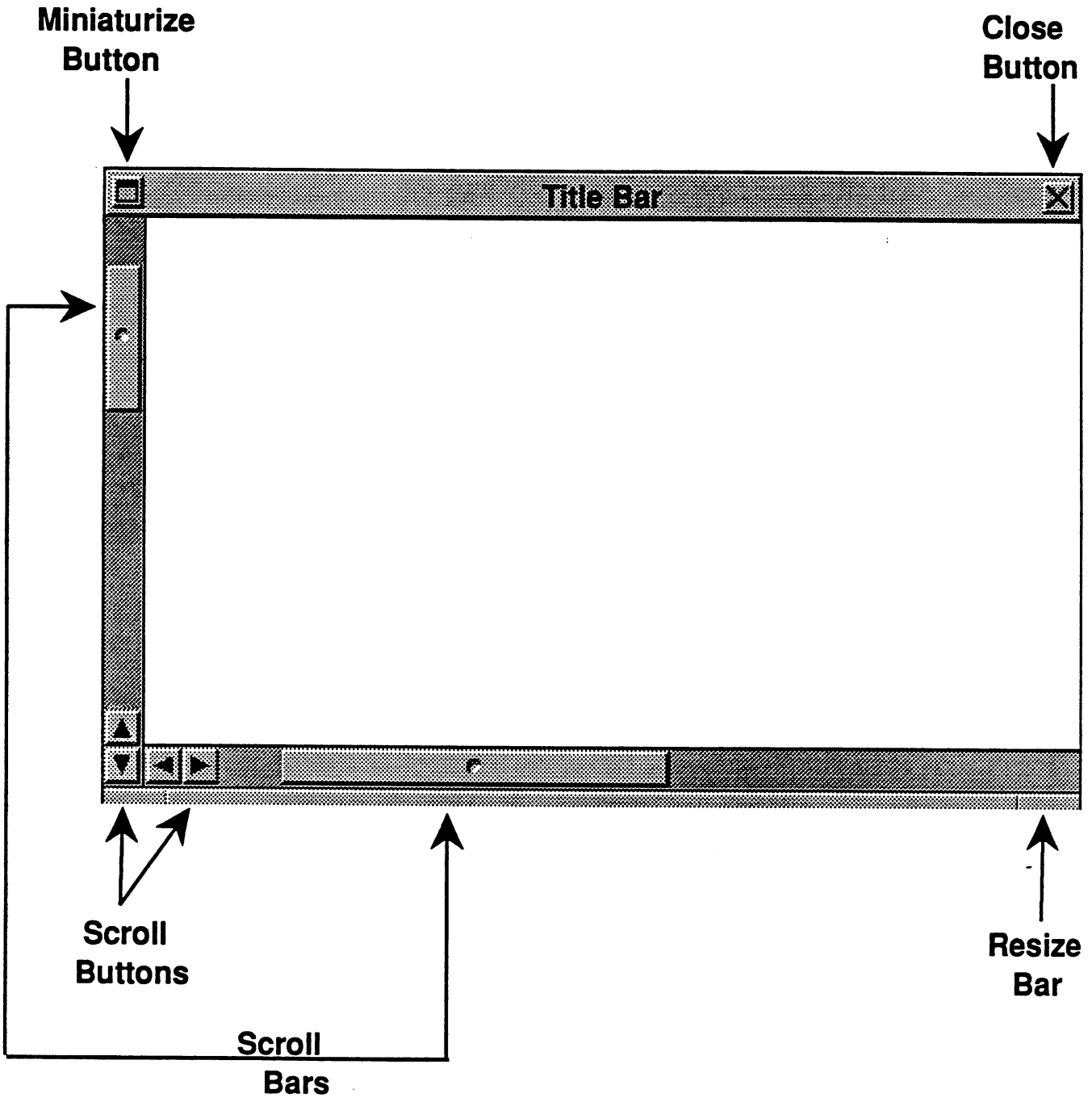
Miniaturizing a standard window: Another way to put a window aside without getting rid of it completely, besides dragging it off the screen, is to miniaturize it. Miniaturizing a window only changes its appearance. A miniaturized window changes into a miniwindow—an icon that appears out of your way at the bottom of the screen. Most standard applications include both a miniaturize button in their title bar and a **Miniaturize** menu command.

Resizing a standard window: It is also easy to change the size of most standard windows (and some panels). Resizable windows can be enlarged or shrunk by relocating the bottom border and/or either the left or right border. Windows are resized by positioning the cursor over the window's resize bar, pressing the primary mouse button, and dragging in the direction desired. As you drag the bar, a dashed line will appear indicating the new size of the window. When the button is released, the window will be resized to match the dashed line. Dragging anywhere between the two marks on the resize bar will change the vertical dimensions of the window. Dragging outside of these marks, on either side, will change the horizontal dimensions, or both dimensions simultaneously. Dragging inward makes the window smaller, outward makes it larger.

Scrolling within a standard window: There are three ways to scroll within standard windows. Pressing a scroll button produces a slow continuous scroll. Selecting and dragging a scroll knob is usually used for larger motions and also produces a continuous scroll. Clicking directly above or below the knob will relocate the knob to the cursor position, producing a jump scroll of the window contents.

Making a standard window the key window: The key window is the standard window or panel that is currently receiving keyboard input. The title bar of the key window will always be black. To make a window the key window, simply click anywhere within its boundaries (other than its buttons). This window will be moved to the front and become the key window.

Standard Window Components



Less Than One Hundred and One Things You Can Do to a Menu

Menus are windows containing lists of commands. They are used to control the actions performed by applications. In addition, several actions can be performed on menus themselves. The following is a list of the most common things users do to and with menus:

Opening a main menu: In most cases a main menu is automatically opened when you launch an application and closed when you quit the application. Usually, only the menu(s) associated with the active application will be visible.

Moving a menu: Any menu (in fact, any window) can be moved by positioning the cursor over the title bar, pressing the primary mouse button, and dragging the cursor and the menu to the desired location.

Making a selection from a main menu: To select a command from the main menu of the active application, simply click the desired command. Menu items appear in one of the following four forms:

> **label only:** Many menu items contain only a functional description of the action that will be performed when the item is selected.
>
> **label and keyboard alternative:** Many menu items also contain a letter indicating a keyboard alternative that can be used to initiate the associated action without the mouse.
>
> **label and "...":** Some menu items, when selected, result in the display of a panel. These items contain "..." following the label.
>
> **label and arrowhead:** Some menu items, when selected, result in the opening of a submenu. These items contain an arrowhead following the label.

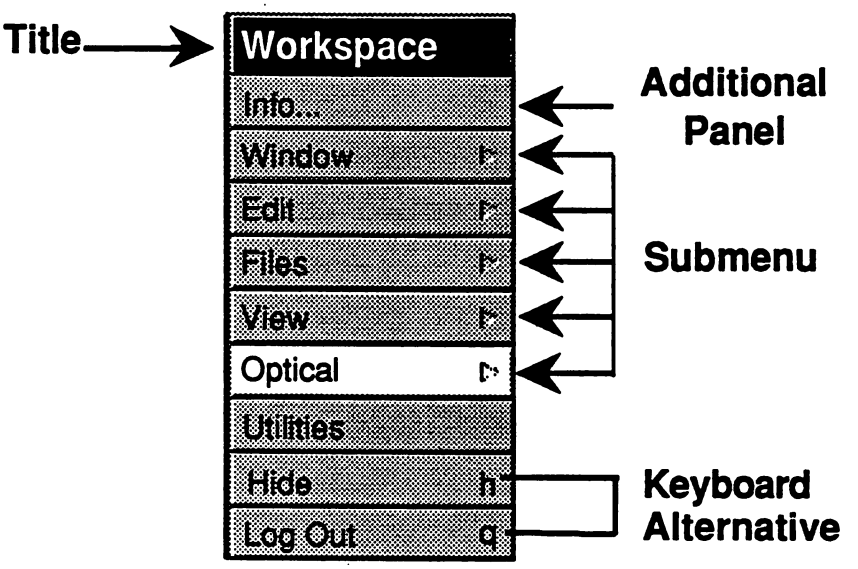
Opening a submenu: To open a submenu, simply position the cursor on any menu item that contains an arrowhead and hold the primary mouse button down. The submenu associated with your selection will appear directly to the right of the main menu. If you release the mouse button at this point the submenu will remain displayed.

Making a selection from a submenu: Whatever menu item is under the cursor when you release the primary mouse button will be selected and executed. Thus, there are two ways to select an item from a submenu. The first is to click the submenu name, wait for the submenu to appear, and click the desired item on the submenu. The second is to press the primary mouse key while the cursor is over the name of the submenu, drag the cursor into the resulting menu, and release the mouse button while the cursor is over the desired selection.

Tearing off a submenu: Submenus can be torn off and moved in much the same way as main menus. Once the submenu is open, simply position the cursor over the menu's title bar, press the primary mouse button and drag the cursor and the menu to the desired location. When you release the mouse button the submenu will grow a close button and remain visible until you close it.

Popping up a menu: For users who believe in economy of motion (i.e., are truly lazy), and don't like having to move the cursor to the menu to make a selection, it is a simple matter to have the main menu associated with the active application pop up at the cursor's current location. One of the **Preferences** options regarding the mouse is **Enable menu button**. If you have selected this option, you can just press down on the secondary mouse button (determined by the **Handedness** option in **Preferences**) to bring up the menu. You can then drag the cursor to the desired selection or submenu and release it. The popup menu will disappear after use.

Standard Menu Components



Less Than One Hundred and One Things You Can Do to an Application

Launching an application: Applications can be launched (executed) in numerous ways:

- double clicking the name in the Browser
- double clicking the icon in the Browser icon well
- double clicking the icon in the dock
- entering a command line in a UNIX shell
- entering the name and pressing **Return** in the Browser **Name Expansion** window
- opening a file associated with the application (double clicking the file's name or icon, or selecting **Open** from the Workspace Manager's **Window** submenu)
- selecting **Dock Info...** from the **Utilities** submenu and checking off the application to be auto started at login

Hiding an application: Almost all applications can be hidden either by selecting the **Hide** menu choice or by using the **Command-h** keyboard alternative. Hiding an application removes all visual objects associated with that application from the screen, including open windows, but excepting the icon. The docked icon associated with a hidden application will be missing its dimples. Applications that are hidden but were not launched from the dock will be marked by their icon at the bottom of the screen.

Unhiding an application: Applications can be unhidden by double clicking their icon in the dock or at the bottom of the screen.

Making an application active: To make an application active, click any visual object associated with that application or double click the associated icon.

Quitting an application: All applications can be exited either by selecting the **Quit** menu choice or by using the **Command-q** keyboard alternative.

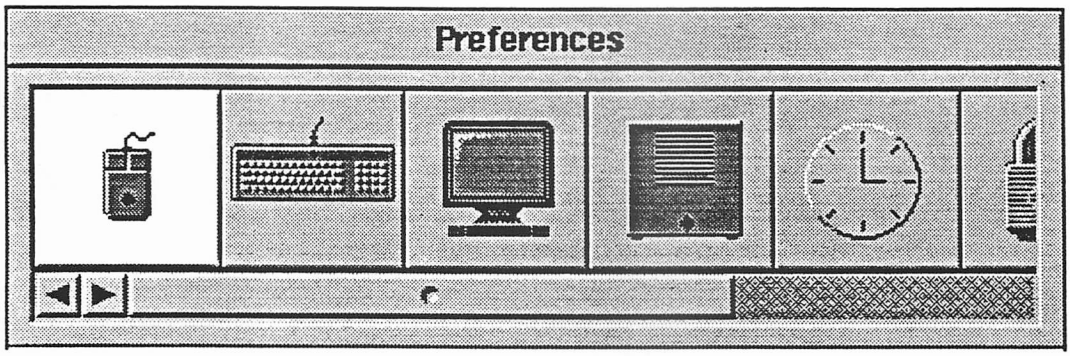
In summary, an application can be in one of five states at any time:

1) Not running
2) Launching
3) Running, inactive, and visible
4) Running, inactive, and hidden
5) Running, active and visible

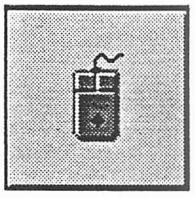
Relationships Between Window and Application Functionality

Applications	Windows
Launch	Open
Hide (icon)	Miniaturize (mini-window)
Quit	Close

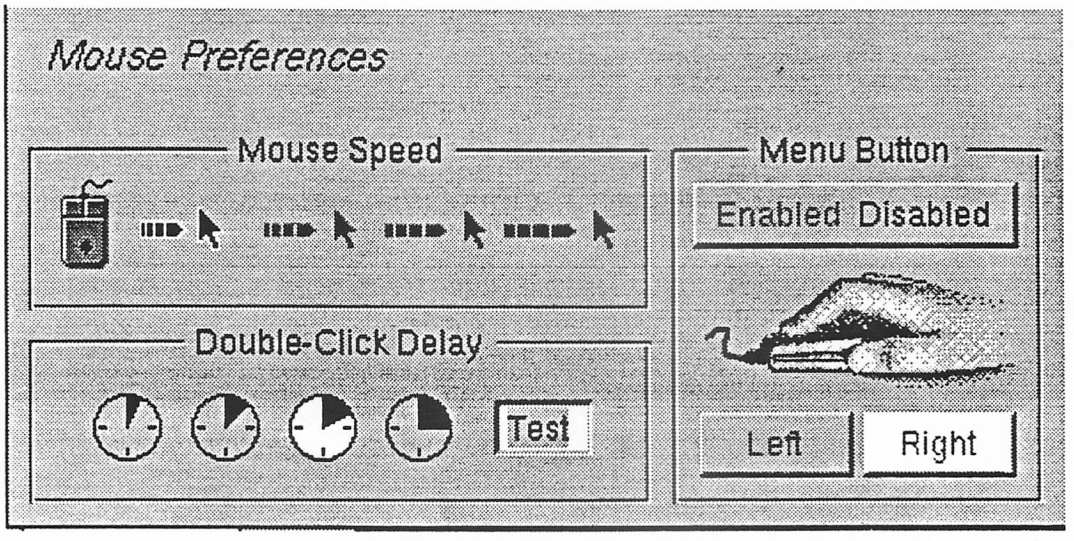
The Preferences Application



The **Preferences** application is used to set several system variables to meet your individual needs. The available options are separated into different components of the system, as described on the following pages.



Mouse

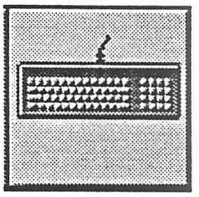


Mouse Speed: Four different speeds are available, with the leftmost button representing the fastest mouse speed.

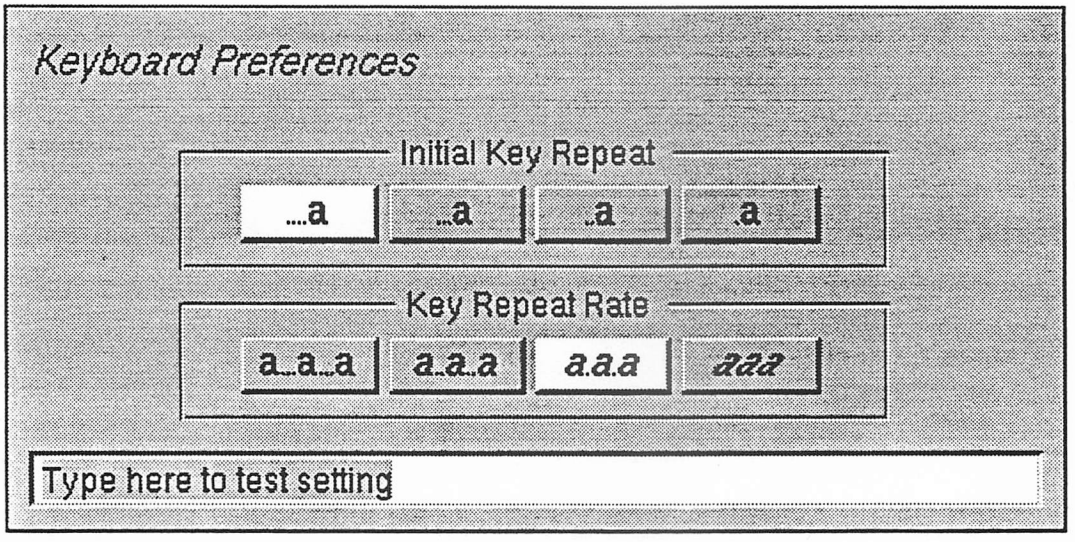
Double-Click Delay: Four different double click intervals are available, with the leftmost button representing the shortest interval (the amount of time allowed between clicks constituting a double click).

Menu Button: When Enabled is selected, pressing the secondary mouse button will display a pop up menu at the current cursor location.

The handedness buttons determine which mouse button will serve as the primary button. The left mouse button is the primary one for right-handedness, the right button for left-handedness.

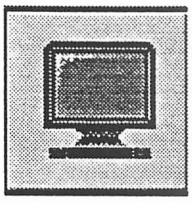


Keyboard

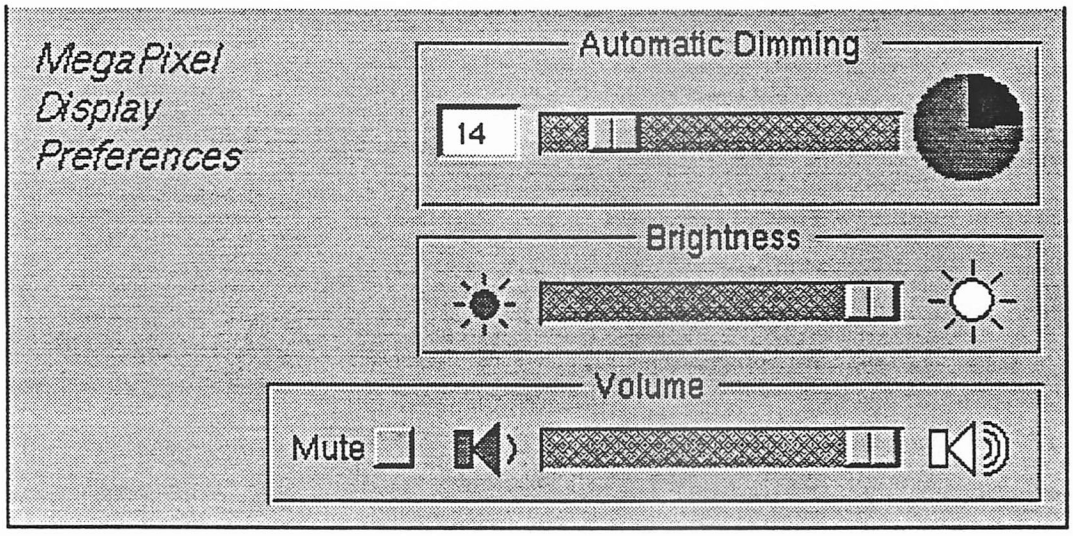


Initial Key Repeat: Sets the amount of time a key must be held down before repeat starts. Leftmost button selects the most amount of time before repeat starts.

Key Repeat Rate: Sets the speed at which keys will repeat when held down. The leftmost button represents the slowest repeat.



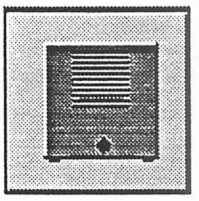
Monitor



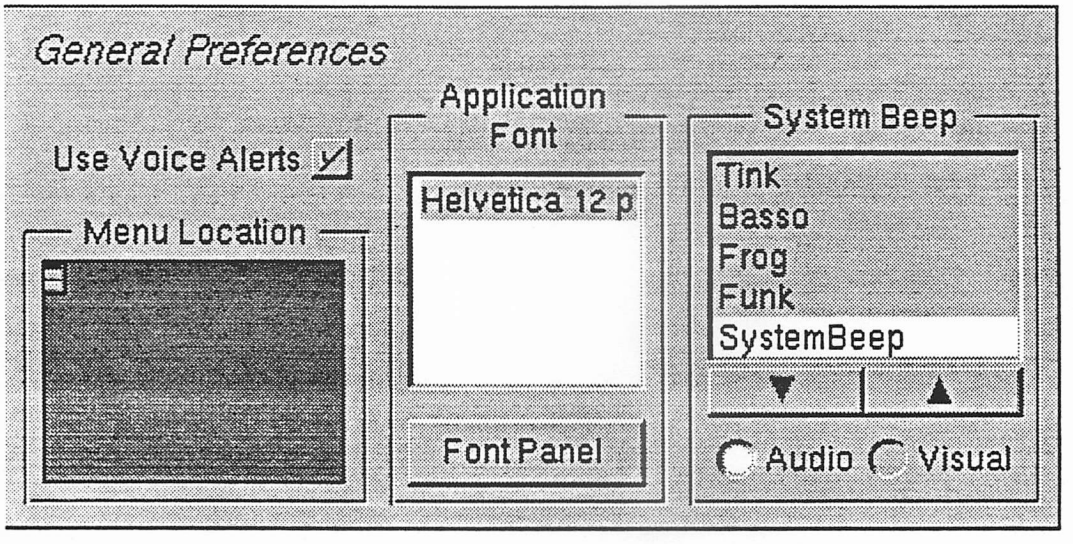
Automatic Dimming: Selects the number of minutes of inactivity before screen will automatically be dimmed.

Brightness: Sets screen brightness, just as pressing the brightness keys do.

Volume: Slider sets volume; button turns muting on or off.



General

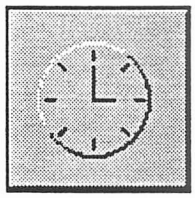


Use Voice Alerts: Allows you to turn on Voice Alerts (verbal messages are up to each individual application; currently, only printing makes use of any).

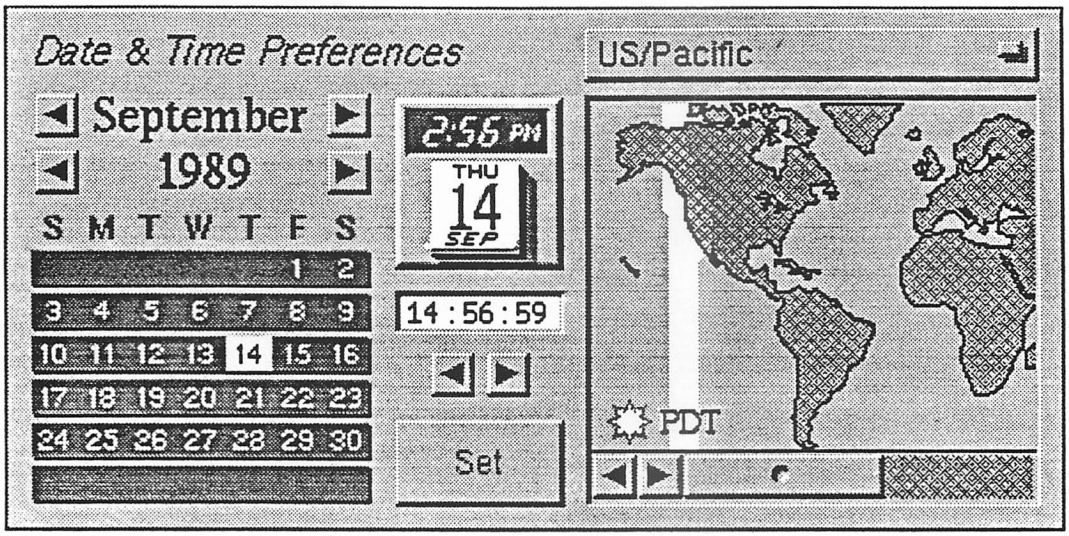
Menu Location: Determines the initial location of the main menu for applications.

Application Font: Sets the default font for applications. The Font Panel button will bring up a font panel from which you can select the application font.

System Beep: Allows you to select the sound used as the system beep. If Visual is selected, the system beep will be represented by the NeXT logo at the top of the dock graying out briefly. The system beep is played or displayed when selected.



Date and Time



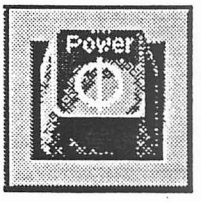
Date: Allows you to change the system date. The arrow buttons scroll select the month and year, while clicking a date in the calendar selects that date.

Clock: Allows you to change the system time, using the arrow buttons (double click the hour, seconds or minutes of the digital clock display before using the buttons).

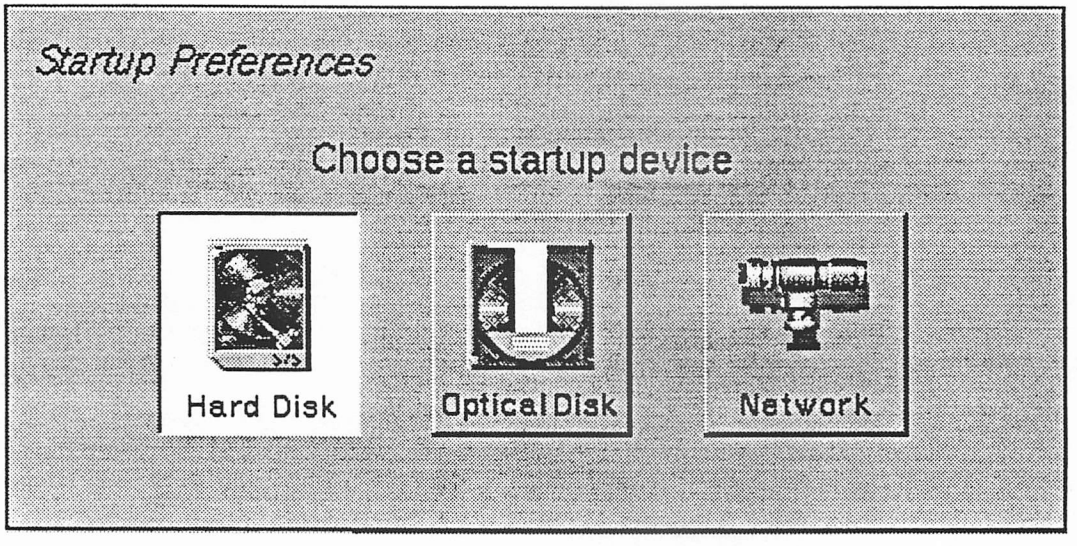
Time Zone: This is where you select the time zone, by clicking the map of the world in the appropriate spot. The pop-up list can be used to further refine your selection.

Password

Changes your **Netinfo** password. If this is unsuccessful, changes your yp password.

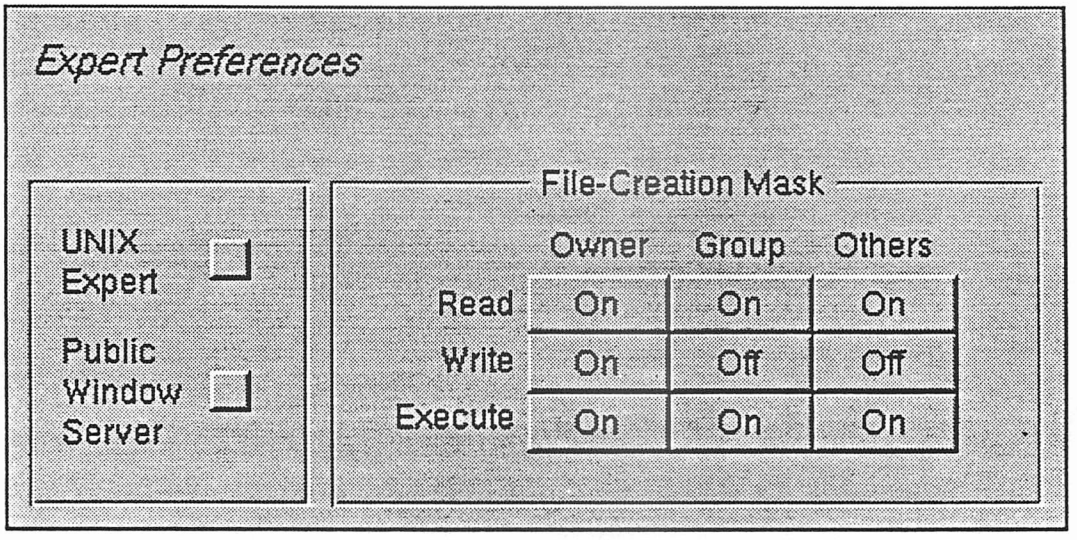


Startup



Allows you to select the boot device (Hard Disk, Optical or Network).

Expert



UNIX Expert: When selected, all files on the system will be visible through the Browser. When not selected, only a subset will be displayed. (See the handout *NeXT Directory Structure* for details.)

Public Window Server: When selected, allows remote connections to your window server.

File-Creation Mask: Determines the default permissions for newly created files.

Edit

Launch the Edit application. Note that it doesn't open a new window for you to work in.

From the **Windows** menu, click **New**. This will cause a new empty edit window to appear on the screen. Notice that it is called **Empty Window** and it's title bar indicates the directory path you would use to find the document. Enter a few line of text by simply typing on the keyboard.

To select a single character:

- Place the I-beam cursor in front of (or behind) the letter you want to select. Click and drag through the letter. It will indicate it's selection by a transparent gray field that will enclose the character. The indication is called *highlighting*.

To select an entire word:

- You can use the same technique. Click at the beginning (or the end) of the word and drag to the opposite end. The entire word will highlight.

- You can simply click at the beginning or end of the word (and not drag) and then—while holding down the shift key—click again at the end of the word. The entire word will highlight. This is called *shift clicking* and allows you to extend a selection.

- You can also use a short cut. Double click anywhere on the word you want to select. The entire word will highlight to show it has been selected.

To select an entire line:

- You can use the click and drag technique (as was used on a single character).

- You can click (not click and drag) at the beginning of the line you wanted to select and while holding down the shift key click at the end of the line. This will select the entire line and cause it to highlight.

- Finally, there is a shortcut for selecting a entire line. Simply triple click on the line you want to select. The entire line from it's beginning to the carriage return that ends it will be selected and will highlight.

To select the entire contents of a window:

- Use the shift clicking technique or the Edit menu's **Select All** menu command.

To change the font attributes of a selection choose the **Font Panel...** choice of the **Font** menu. A panel will appear. This panel will list all the fonts and styles of the fonts that are available on your particular machine. Choose a **Family**, **Typeface** and **Size** combination. To preview what your choice will look like, click **Preview**. The font panel's text field will change to reflect the choices you made, leaving your document's

text unchanged. To apply those choices to the text in your document window click **Set** or hit Return.

Notice that changing the selection of a single character changes the entire window of text.

These text handling characteristics demonstrated in Edit are typical of most text based application on the NeXT Computer System.

To access applications like Webster, the Digital Librarian, or Quotations, select a word and click the **Request** menu. To look up the word in Webster, simply click **Define in Webster** (or use the *command key equivalent* **Command =**). Webster will launch, and the word will be defined

Finally, to save the file you've been working with, click in the Edit window, then choose **Save** from Edit's **Window** menu. You will be greeted with a *Save panel*, a mini-browser that shows the files in the file system. It will open in you home directory. Type a file name in the Save panel's text field and hit return. If the filename already exists you will be prompted by the system for how to proceed.

Hidden Applications, Windows and Mini-windows

Introduction

Applications are launched and can be hidden. Windows are opened and closed, miniaturized and de-miniaturized. The NeXT environment provides a friendly and unique way to use these entities to help you work more effectively.

Applications and Windows

- **Launching an application:** Launch the **Edit** application from the NextApps Directory. Note that a new window is not automatically opened but the icon turns light gray and then appears on the bottom of your screen. The menu for the **Edit** application is in the upper left corner of the screen. The icon at the bottom indicates an active application. Note how long it takes to launch an application.

  Compare this with launching an application from the dock (**Webster's** would be a good choice). Note there are three dots (*ellipsis)* on the icon before it is launched. Is there any difference in the launch time? There shouldn't be. Notice the icon lightens briefly, and the application's menu appears. This time a window also appears, but there is no icon at the bottom of the screen. Instead, the icon on the dock has lost the ellipsis.

  Running applications are indicated in one of two ways: If the application is launched from the dock, the docked icon will lose the ellipsis. Other running applications are indicated by the appearance of the application icon at the bottom of the screen.

- **Selecting a key window:** Double click on the **Edit** icon to make the application active with its menu appearing at the upper left. Notice that the **Webster** menu has been replaced with the **Edit** menu, but the **Webster** window stays on the screen (with a gray title bar). Now click on the **Webster** window to make it the "key" window. Notice the menu changes again, indicating the **Webster** application is active and the title bar on the window turns black, indicating it is the key window.

  The key window is always associated with the active application.

- **Quitting an application:** You can quit an application whether or not a window is open. Choose **Quit** from the **Webster** menu. Notice the ellipsis re-appear on the icon.

- **Opening and closing windows:** Double click the **Edit** icon and open a new window by choosing **New** from the **Window** menu. Notice the other options on this menu, especially **Save, Miniaturize** and **Close.** Note the name of the window and the button with an X in the upper right corner. Close the window by clicking this button.

A window can be opened and closed from the Window menu. A window can also be closed by clicking the close button in the upper right corner.

- **Saving the contents of a window:** Open a new window again. This time enter a few lines of text by simply typing on the keyboard. Notice the X in the button has changed to a "broken" character. Save the text by clicking either **Save** or **Save As...** on the **Window** menu. Since it is an untitled window, the **Save** panel appears, allowing you to name the document, and save it under the directory you specify. It defaults to your home directory.

  Once you have saved the document, notice the name in the title bar changes, indicating the directory path you would use to find the document. Use the menu choice to close the window and open it again, this time using **Open** from the menu. Add more text to the document, and this time click the close button. A panel appears giving you the option to cancel or update and overwrite the original file. After the document is saved, the window is also closed. The **Quit** menu selection would operate in the same way, offering you an option to cancel or save the file before quitting the application.

There are four ways to save the contents of a window: 1. The Save option from the menu will automatically update the file currently being worked on. 2. Save As... gives you the option to save the file in a different directory, under another name or to overwrite the existing file. Both of these methods leave the window open. 3. Using the close button gives you a choice of canceling, or updating the file and closing the window. 4. The Quit menu selection notifies you of any open windows and offers the option to save them before closing the windows and quitting the application.

Multiple Windows

It is often convenient to have multiple windows open at the same time. You might want to refer back and forth between windows, or copy and paste text between windows.

- **Copying and pasting between windows:** Using the **Edit** menu, this time open two windows. Use the **Window** menu and open the following two documents:

  /NextDeveloper/Examples/Clock/AnalogClock.h
  /NextDeveloper/Examples/Clock/AnalogClock.m

  These contain header information and the main body for the AnalogClock application. Make the AnalogClock.h window the key window. Position the I-beam cursor in front of a line of text and click the mouse. Click and drag to select several lines of text. Once the text is highlighted, release the mouse and choose **Copy** from the **Edit** menu. Make the AnalogClock.m window key, position the cursor in the window and click the mouse button to place the I-beam as the insertion point. Choose **Paste** from the menu to insert the copied portion into the text. Don't save the changes at this time.

- **Miniaturizing windows:** When you are using several windows at one time, you might want to keep the windows handy, but not clutter the workspace. Select the AnalogClock.m window and move it as far off the screen as possible, by dragging the title bar. This is one way to clear the screen.

  A more effective way to use the windows and screen space is to miniaturize the windows. The button in the upper left corner of the window is the miniaturize button. Click it in the AnalogClock.h window. The window shrinks and is placed at the bottom of the screen, near the **Edit** icon. How can you distinguish between the miniature window and the application icon? Right, the mini-window has a black title bar, just like a big window has a title bar. Miniaturize the AnalogClock.m window. Notice that the windows are different colors.

  *In the **Edit** application, a dark gray mini-window indicates that the window has been changed, but is not saved.*

  Bring the AnalogClock.h window up to full size by double clicking the mini-window. Notice that, unlike an application icon, it does not turn light gray. The window immediately expands because we didn't close the window, and therefore have no need to re-open it. There is a noticeable difference between the time to it takes to launch an application and the time it takes to enlarge a mini-window, isn't there?

Windows can be miniaturized by clicking the button in the upper left corner. They can be expanded again by double clicking the mini-window.

See what happens if you miniaturize the **Directory Browser.** What is the name on the title bar?

- **Hiding applications:** Miniaturize all of the windows on your screen. Launch **Webster.** Notice the window doesn't have a miniaturize button. Instead, choose the **Hide** option from the menu. Notice that the window and menu go away, but the icon stays on the screen. You can look at the docked icon and see that it is still running, because the ellipsis is missing.

  Make one of the **Edit** windows large. Choose **Hide** from the **Edit** menu. What happens to the big window? How about the mini-windows? Right, all the windows and the menu are gone, but the **Edit** icon stays on the bottom of the screen, indicating it is still running.

  Double click the **Edit** icon. The windows all come back to the same position they were in before they were hidden. Hiding an application that you expect to work on later can save you lots of time, compared to closing all the windows, and quitting an application, then re-launching the application and opening up the windows you wanted to use.

  Hide all the applications that you have active on the screen, including the **Workspace Manager.** In order to hide it, you need to make it active. You could do this by either enlarging the **Directory Browser** window, which brings up the **Workspace Manager** menu, or by double clicking the **NeXT** icon at the top of the dock. Quit the **Edit** application without saving any changes.

  Hiding an application removes everything from view that is relative to the application, except for the application icon, which may be on the dock or at the bottom of the screen. Everything can be brought back into view by double clicking the icon. Hiding an application is relative only to the appearance of the screen. It has no connection to saving a window, closing a window, or miniaturizing a window.

Summary

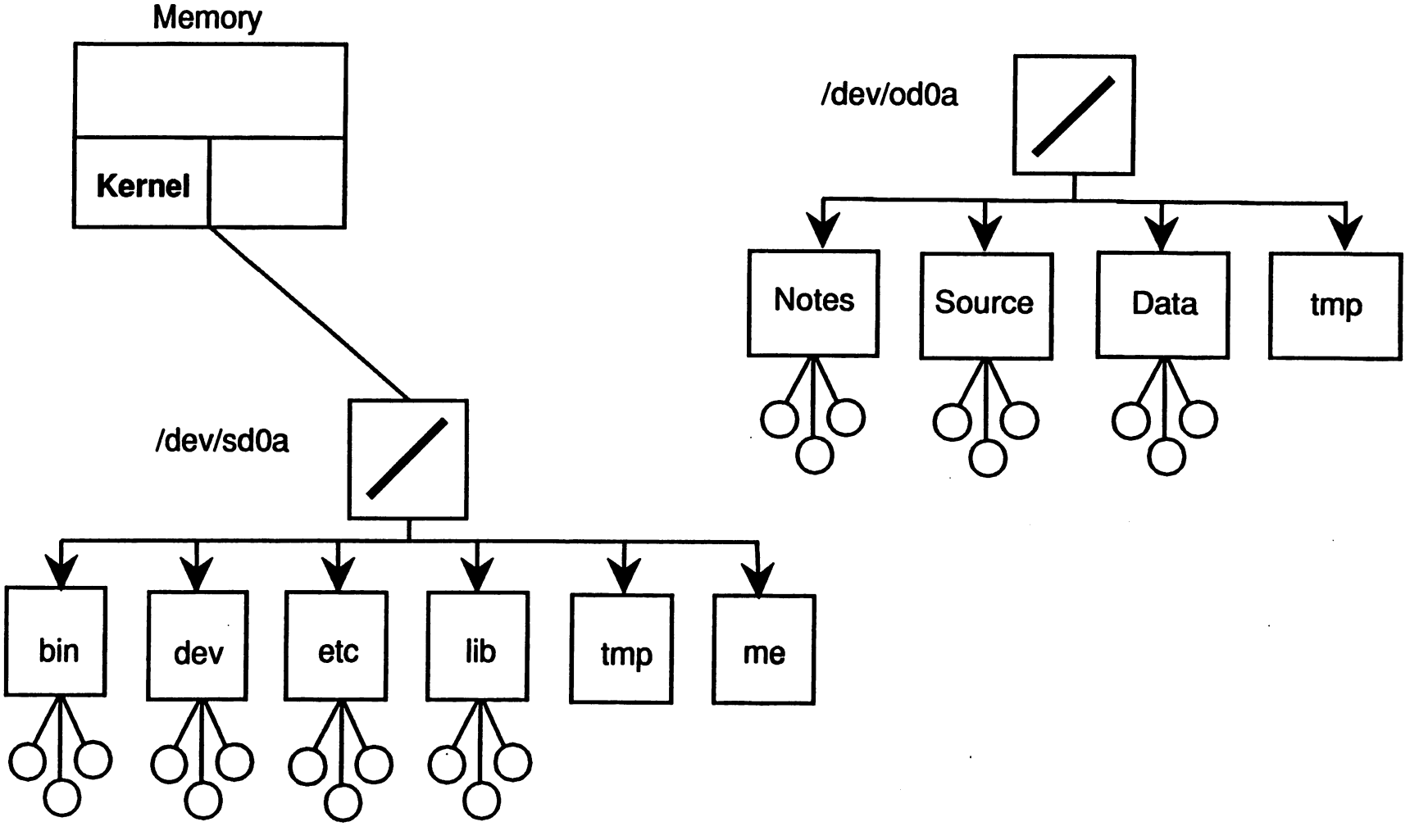
The operations which apply to applications are launch, quit, hide and expose. Windows are associated with applications, just as a menu or a panel might be associated with an application. Therefore, performing one of these operations on an application will affect the windows associated with that application. Open, close, miniaturize and expand relate only to windows, and therefore have no effect upon an application.

During a typical working session, you will probably want to launch several applications, and keep them running, rather than launch, quit and re-launch. This will allow you to move freely between different windows and applications. You will know which applications are running by the state of the icon on the screen. The windows that you use frequently can be kept handy by miniaturizing them or you can clear the screen by hiding those applications you will be referring to less frequently.

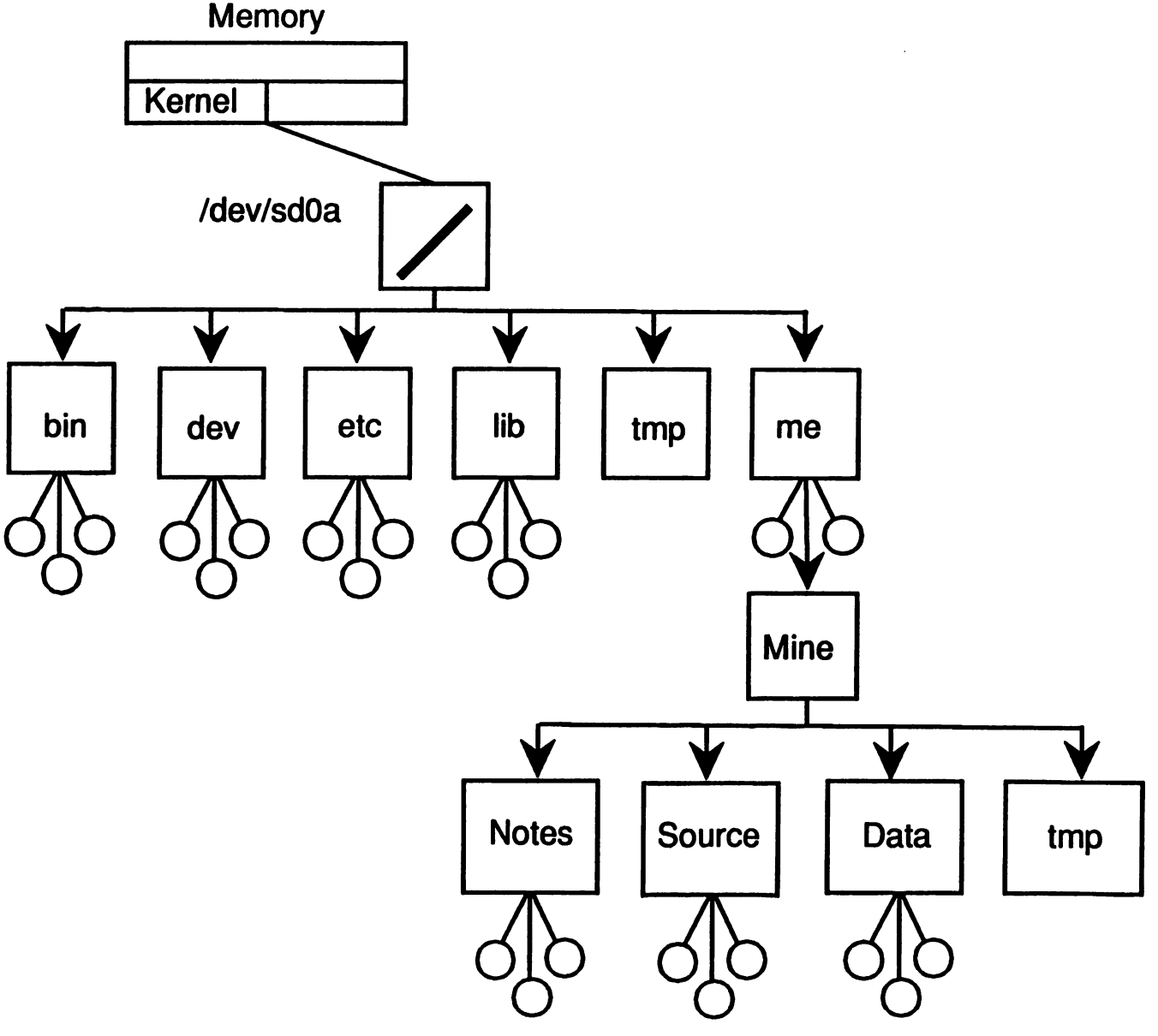
Hints for Using a NeXT Computer

- Take advantage of multi-tasking capabilities, avoid the launch and quit
- Use inter-application cut and paste
- Use inter-application messaging
- Stop worrying about available memory, no file or program is too large
- Program it, it's fun

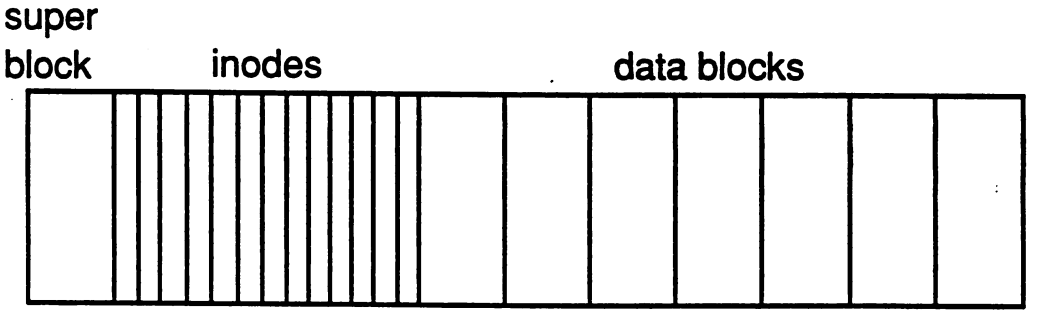
Mounting Filesystems 1



Mounting Filesystems 2



File System Components

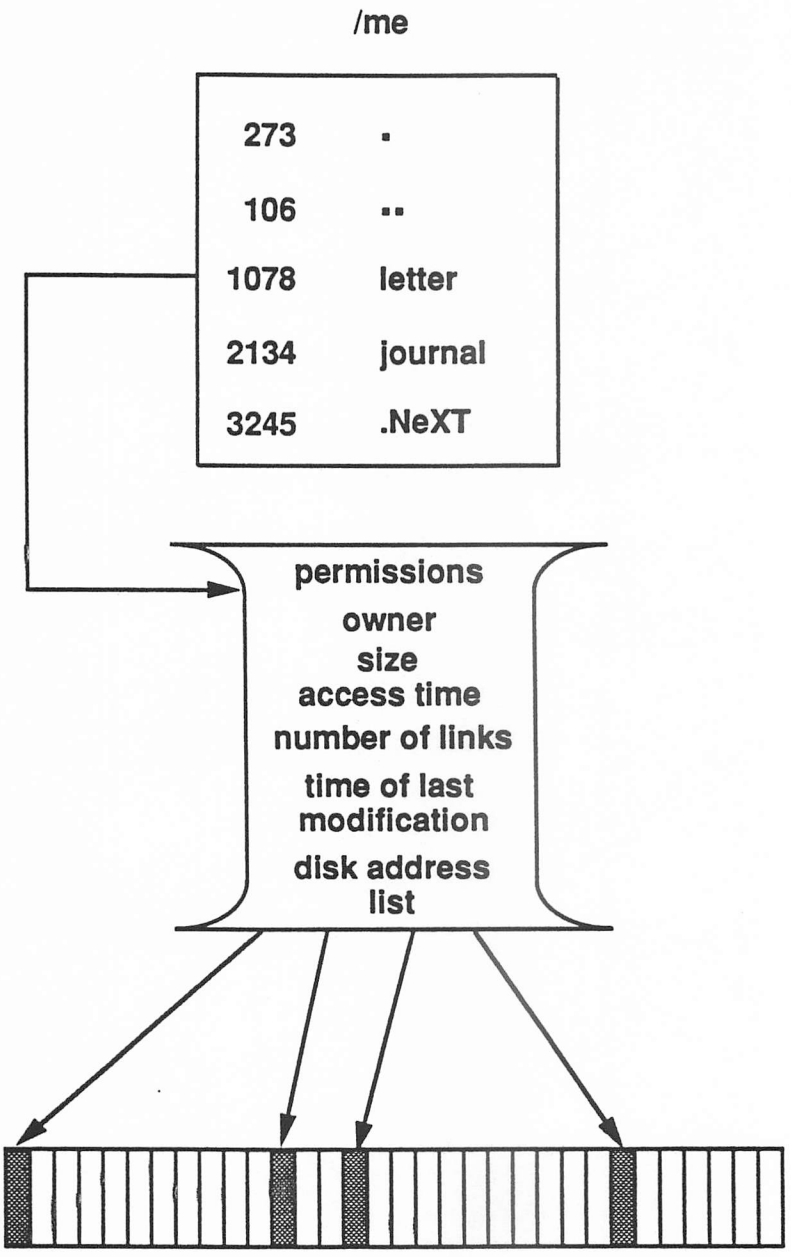


Structure of a Directory File

/me

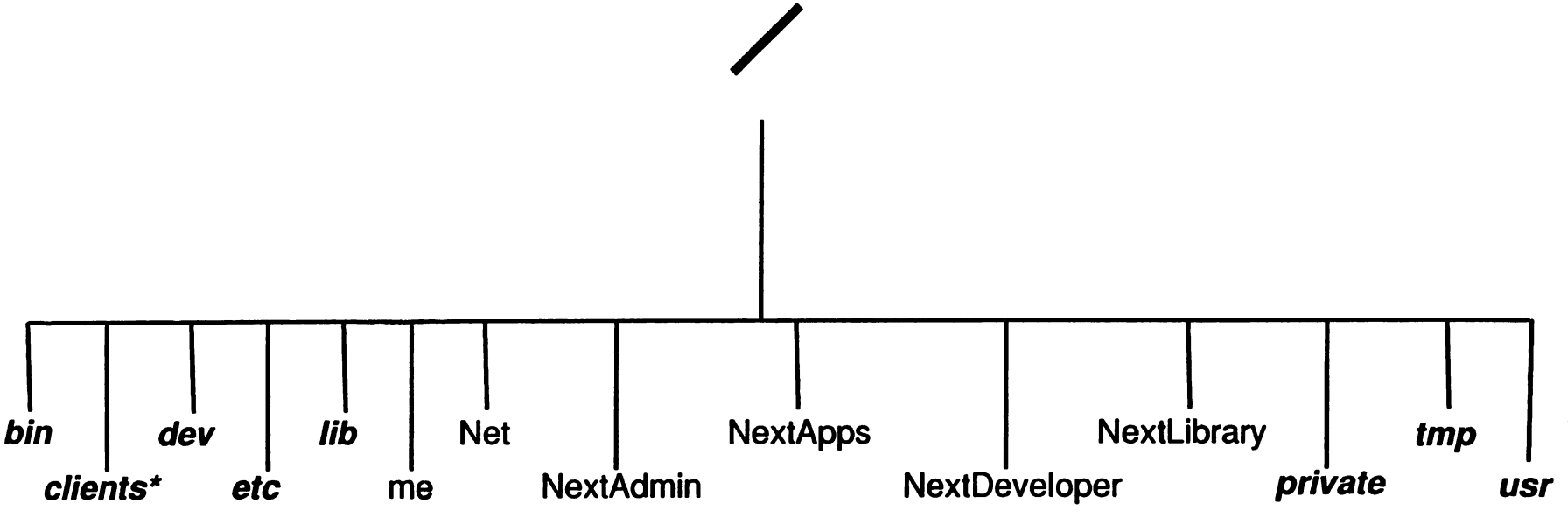
273	.
106	..
1078	letter
2134	journal
3245	.NeXT

Three Parts of a File



A Portion of a Disk

NeXT Directory Structure



* only on a 660MB SCSI system

directory only appears when UNIX Expert is selected in Preferences

Shell Tubes

There will likely be times when you want to access the UNIX shell directly. Two applications are provided for this purpose. They are both located in the directory **/NextApps**, and are frequently found populating the docks of reformed UNIX users.

Using the Shell application: This application accesses the shell specified in your password entry and supports scrolling, editing, and printing of its window. Unfortunately, this application currently does not provide the terminal emulation to support screen-oriented utilities like **vi**.

Using the Terminal application: This application also accesses the shell specified in your password entry and provides vt100 terminal emulation to support screen-oriented utilities like **vi**. Unfortunately, this application currently does not provide window scrolling or editing.

Accessing the UNIX Shell on a NeXT Computer

1 **Launching a Terminal application from the Browser:** Use the Browser to access the directory **/NextApps** and double click **Terminal.**

2 **Running shell commands:** At this point you can do anything you can do in a common shell. For now, just enter the commands:

```
date
ps ax
```

3 **Using more:** Enter the following:

```
more /etc/termcap
```

The **more** command is used to display the contents of a file on the screen, one page at a time (the file **/etc/termcap** was chosen because of its length; it's a file containing terminal capabilities).

At this point, the utility is waiting for instructions on what to do next. Press the **Space bar** to display the next screenful of the file. Now press **Return** to display the next line. Again. Now, press **q** to exit the **more** utility.

As you can see, the more utility functions quite well from the **Terminal** application.

4 **Using the Shell application:** Let's repeat this sequence of commands using the **Shell** application. First, launch **Shell** and enter the commands:

```
date
ps ax
```

5 **Using the Shell's scroller:** Use the scroller to view the output of your **ps** listing. Sort of makes the **more** command obsolete, doesn't it?

6 **Jumping back to the present:** Use the scroller to move somewhere towards the beginning of your Shell's stored text. Now, to jump to the end of your stored text (back to the present), press **Command-j**.

7 **Re-executing a previous command:** Use the scroll bar to scroll up until the **date** command line becomes visible. Select this line by positioning the cursor over the first character in it, pressing down on the primary mouse button, dragging to the last character in the line, and releasing the button. Now press **Command-c** and then **Command-v**. To see what just happened, press **Command-j**. Pretty neat, isn't it? At this point, you can add to the command line or edit it. To execute the line as is, press **Return**.

8 **Using more from Shell:** Now, try out a command that uses full screen capabilities:

```
more /etc/termcap
```

Press the **Space bar**. That seems to work fairly well; the next screen of file contents is displayed. Now, press **Return** a few times. Oops! **Shell** doesn't perform terminal emulation, and couldn't quite handle that operation. Press **q** to exit **more**.

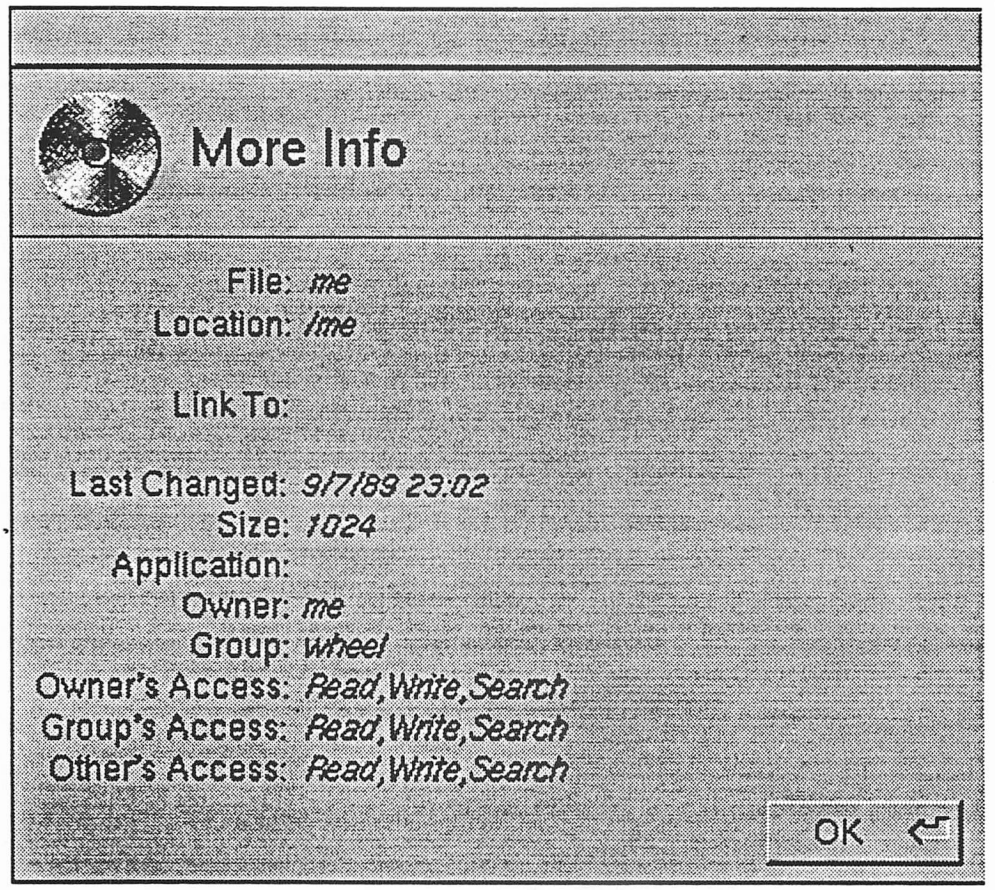
The moral of the story is, use **Terminal** for UNIX tasks that require terminal emulation and **Shell** for everything else.

9 **Emptying history:** As with normal C-Shell history, the **Shell** application's stored text becomes a burden as its size increases. To empty the **Shell**'s stored text use the command **Command-k**.

File Permissions

Examining Permissions

Stored with each file on the system is a set of permissions which dictate what type of access is available to users. To examine these permissions, highlight the file name in the Browser and select the menu option **More Info...** under **Files** (the keyboard alternative for this is **Command-?**). The resulting panel includes quite a bit of information about the file:



Owner: This is the user name associated with the file.

Group: This is the group name associated with the file.

Owner's Access: This field indicates what actions can be performed on the file by the owner of the file (in this case, **me**).

Group's Access: This field indicates what access actions can be performed on the file by the members of the user group associated with the file (in this case, **wheel**).

Other's Access: This field indicates what actions can be performed on the file by all other users on the system.

Access granted by file permissions:

Read: Allows the user to examine the contents of a file.

Write: Allows a user to change the contents of the file.

Execute: Allows the user to launch the contents of the file (assuming it is an executable file).

Access granted by directory permissions:

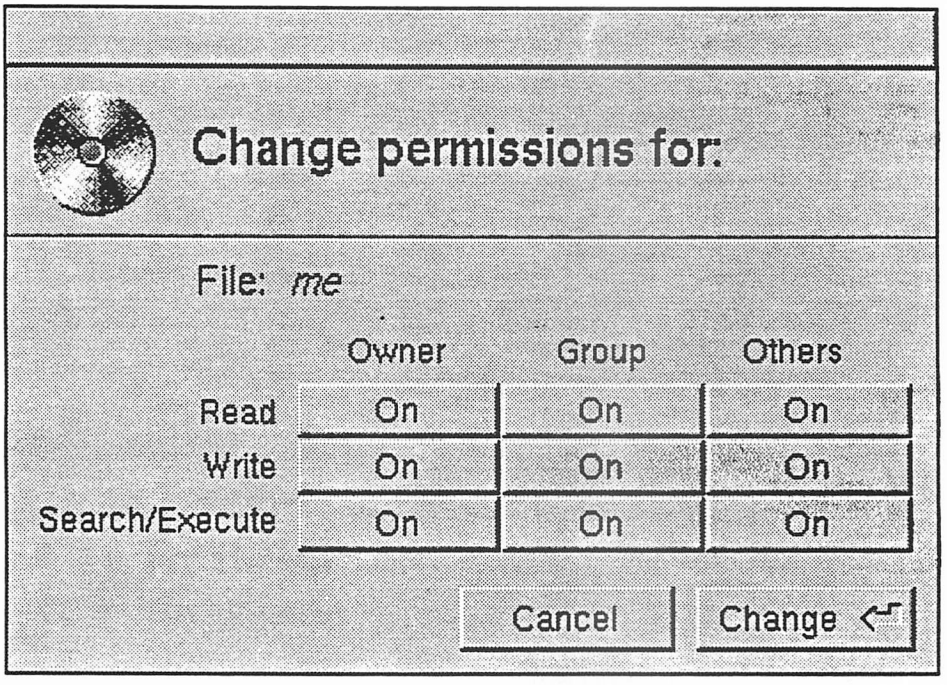
Read: Allows the user to display the contents of a directory.

Write: Allows a user to add a file to or remove a file from the directory (regardless of the permissions set on the file itself).

Search: Allows the user to select the directory in the Browser (in a UNIX shell, allows the user to `cd` into the directory or use it in a search path).

Changing Permissions

To modify the access permissions for a file or directory, highlight the file name in the Browser and select the **Protect...** option from the **Files** submenu of the Workspace Manager (only the owner of the file or the super user can change the access permissions). A panel will appear:



A button is provided for each of the three types of access permissions for each of the three classes of users. Clicking a button will change its value, of which there are three possibilites: **On** (that specific permission is granted for that specific class of user), **Off** (that permission denied) or **No Change** (leave the permission as it was set before you began the operation). Once the permissions are set the way you would like, click **Change** to have the changes take effect.

Setting Default Permissions

Whenever a file or directory is created, a default set of permissions are assigned. These defaults are set in the **Preferences** application under the **Unix Expert** section.

Changing a File's Group Affiliation

When a file or directory is initially created, it inherits its group affiliation from its parent directory. It is entirely possible that you may want to change the group associated with a given file at a later date. Currently, the only way to do this is with a UNIX utility.

To change a file's group, launch either **Terminal** or **Shell** and enter

```
chgrp groupname filename
```

where *groupname* is replaced by the name of the group you wish associated with the file *filename*.

Changing a File's Owner

There are times when it may be prudent to change the owner of a file, such as when one user is "giving" the file to another or when a user account has been deleted. To perform this task you must again use a UNIX utility, with the aditional restriction that only the super user can change a file's ownership.

To change the owner of a file, launch either **Terminal** or **Shell**, use **su** to become root, and enter

```
chown username filename
```

where *username* is replaced by the name of the user account you wish associated with the file *filename*.

NAME

su – substitute user id temporarily

SYNOPSIS

su [–f] [–] [userid]

DESCRIPTION

Su demands the password of the specified *userid*, and if it is given, changes to that *userid* and invokes the Shell *sh*(1) or *csh*(1) without changing the current directory. The user environment is unchanged except for HOME and SHELL, which are taken from the password file for the user being substituted (see *environ*(7)). The new user ID stays in force until the Shell exits.

If no *userid* is specified, "root" is assumed. To .I su to "root", you must supply the root password To remind the super-user of his responsibilities, the Shell substitutes '#' for its usual prompt.

The –f option prevents *csh*(1) from executing the .cshrc file; thus making *su* start up faster.

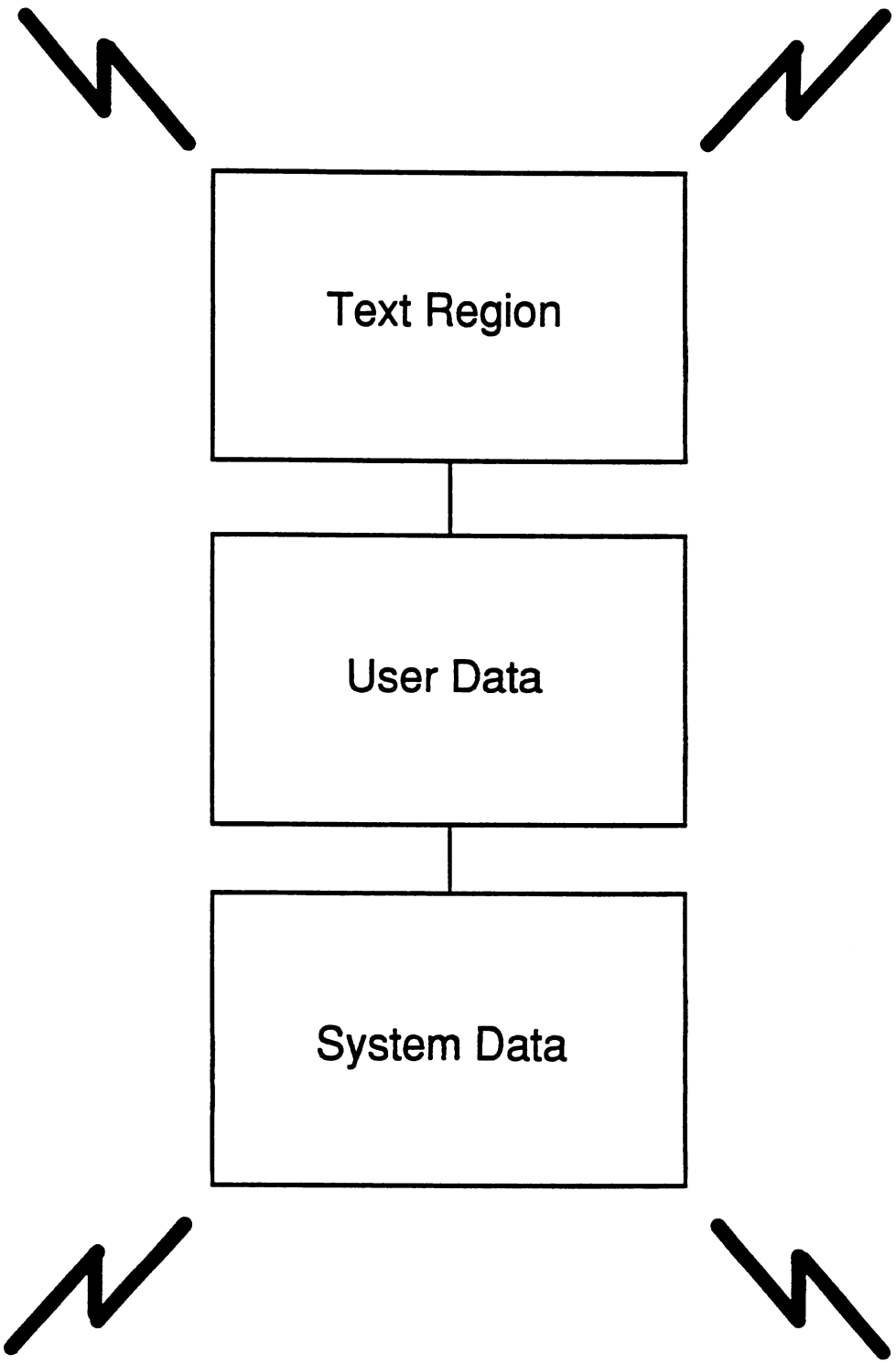
The – option simulates a full login.

An alternative implementation of *su* is in **/bin/su.wheel**. This version, which is shipped without the setuid-root bit turned on, allows users in the "wheel" group (group 0) to *su* to "root" using either their own password or the root password. This version offers more convenience, but if your site is concerned about the increased potential for security problems, you should continue to use the standard version of *su*.

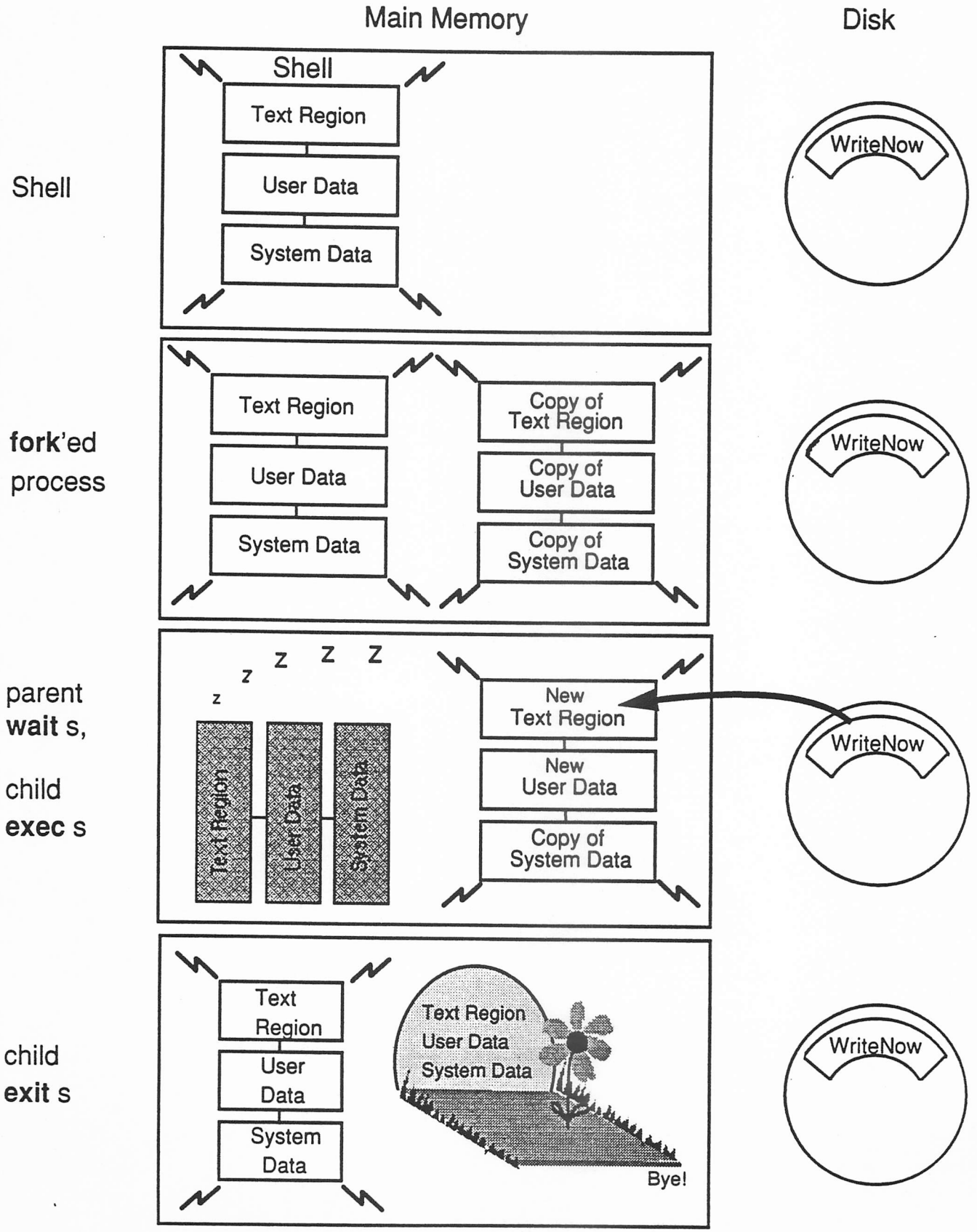
SEE ALSO

sh(1), csh(1)

Image of a Process



Life Cycle of a Process



Process Monitoring and Control

Examining Basic Process Information

To begin, launch two applications—**Shell** and **Terminal**. Make the **Shell** window active, and enter

```
ps
```

This command (process status) will display information about processes running on your system. What you see should be something similar to this:

```
PID TT STAT  TIME COMMAND
430 p1 S     0:00 - (csh)
392 p2 S     0:00 - (csh)
```

The fields displayed are:

PID	Process IDentifier. The number by which a process is uniquely identified.
TT	Controlling terminal (or TTY). The port (or pseudo port) to which the process is attached.
STAT	The STATe of the process. This field is made up of four characters (2 or 3 of which will probably be blank) indicating such things as a Sleeping process, a process that is sWapped out, etc.
TIME	The amount of cpu TIME used by the process.
COMMAND	An indication of the COMMAND running (**csh** is the C-Shell, which is run by both the **Shell** and **Terminal** applications).

Examining a Long Process Listing

To get a long listing of your processes, enter

```
ps -l
```

(That's the letter l, not the number 1.)

This time, the output includes more information than you will probably care to deal with:

```
  F  UID   PID  PPID CP PRI NI ADDR VSIZE RSIZE WCHAN STAT TT  TIME COMMAND
  1  287   430   429  0   0  0    0  840K  304K     0 SW   p1  0:00 - (csh)
201  287   392   391  0   0  0    0  832K  288K     0 S    p2  0:00 - (csh)
```

The new fields displayed in this output are:

F	Flags. A series of digits indicating which flags are associated with the process (if you're curious about the possibilities, check the man page for ps).
UID	The User ID number associated with the process.
PPID	The Parent Process IDentifier. This number identifies the parent of the process.
CP	CPu utilization factor.
PRI	The process' PRIority level.
NI	The process' NIce increment.
ADDR	The swap ADDRess of the process.
VSIZE	Virtual SIZE of the process (in 1k units).
RSIZE	Real SIZE of the process (in 1k units).
WCHAN	Address of the event on which the process is waiting.

Whew—as you can tell, there's a lot of information in this output, most of which you will not usually need.

Examining Process Information for All Processes

Now, to examine information about all the processes, enter

```
ps -ax
```

Somewhere in the output will be a line identifying the process for the **Terminal** application, looking similar to this:

```
429 ?  SW    0:02 /NextApps/Terminal -MachLaunch 18 307
```

Trying to Kill a Process

Now, make note of the PID for the **Terminal** process, and attempt to terminate it with:

```
kill 429
```

replacing *429* with the number of your **Terminal** process.

Oops. You got an error message, didn't you? Something like

```
429: Not owner
```

What might you do to get around this?

Killing a Process

Right, use **su** to become root (root can do anything).

Now, once again enter

```
ps -l
```

to identify the processes currently runnning.

```
     F  UID   PID  PPID CP PRI NI ADDR VSIZE RSIZE WCHAN STAT TT  TIME COMMAND
400201    0   546   533  0   0  0    0  832K  288K     0 S    p2  0:01 -u (csh
     1    0   548   546  0   0  0    0  728K  152K     0 R    p2  0:00 ps -l
```

Notice that the UID associated with these processes is 0. Also notice that the parent of the current csh process is the csh process started by **Shell** that you were examining earlier. Apparently, you have created a new process with a user id of 0.

Now that the current process has a 0 UID, you can once again attempt to kill that pesky **Terminal** process:

```
kill 429
```

Again, replace 429 with the appropriate process number (it's the same process number as in the earlier **kill** command).

Success! The Terminal application very considerately died for you.

Determining Guidelines for Killing Processes

Exactly how did you get **Terminal** to die? The **kill** utility sends a signal to a process that usually results in that process terminating. If your first kill attempt does not succeed, you can try a couple of variations. **kill -1** (this time, that *is* the number 1) will send a hangup signal, which is more likely to kill a process. If that doesn't work, you can try **kill -9**, which sends a signal that cannot be trapped and should kill just about anything. Always attempt a plain **kill** first rather than a **kill -9**. The latter does not give the process a chance to clean up after itself, whereas a plain **kill** is more graceful.

It is also important to kill processes in a specific order. Not to be too gruesome about it, but *kill the children first*. There are a couple of reasons for this. The first is that the purpose of killing a process is to fix a problem, and killing a process that is high in the hierarchy may be a case of overkill. The second is that, in rare instances, killing a process may not successfully terminate all of the children (and grandchildren...). You may even end up with a defunct process.

In this example, it was necessary for you to become the super user before you could successfully terminate the process. This is not always true, and it is a good idea *not* to become root unless you must. If you accidentally try to kill the wrong process as a regular user, you will probably receive an error message. If you try to kill the wrong process as root, it will probably terminate, with possibly unfavorable results.

Dealing with Defunct Processes

A defunct process is created when a process attempts to exit and its parent process has not waited for it (poor thing). These processes will be displayed in the **Command** column of a **ps** listing as

<defunct>

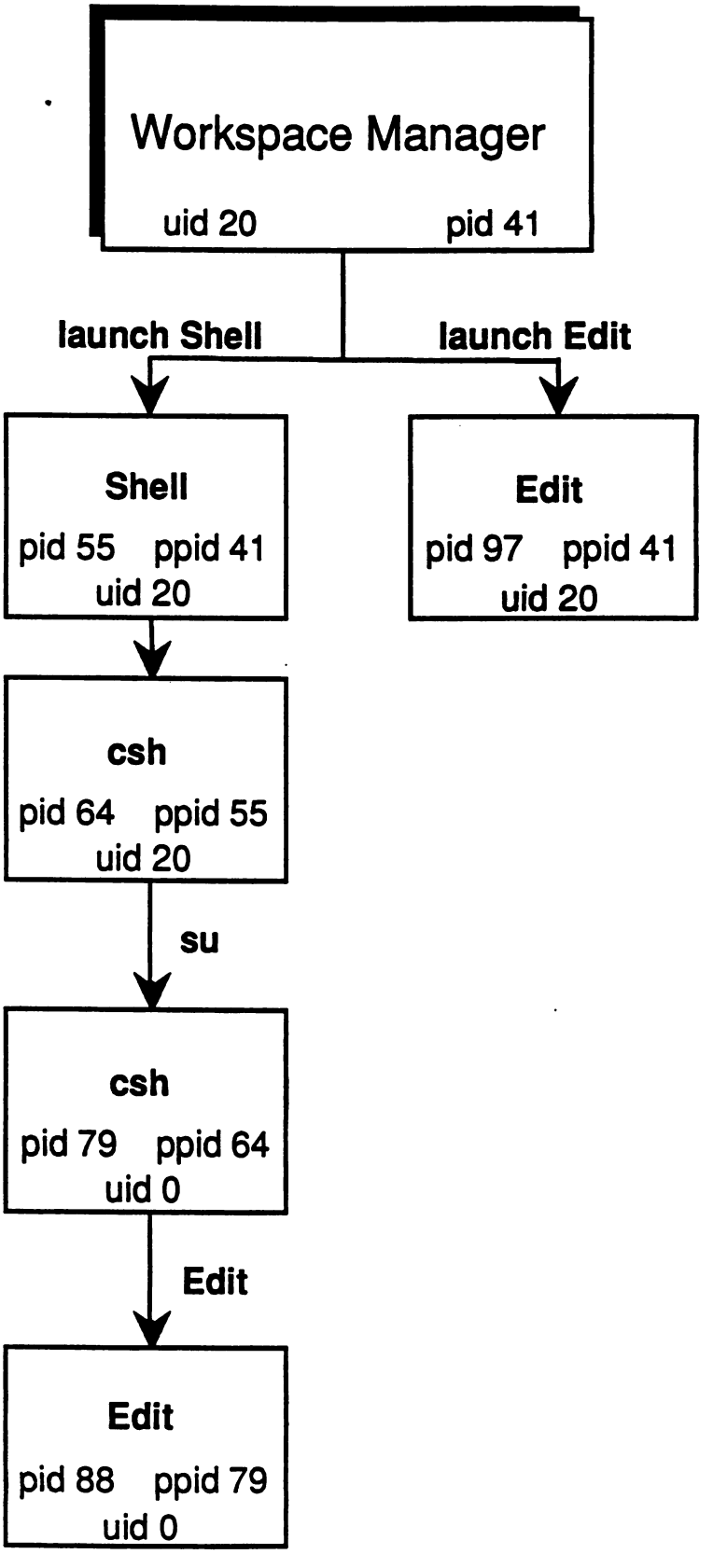
Once you have discovered a defunct process, you have several choices of action. The first is just to ignore it, it will probably go away after a while. If, however, the process is using up system resources (which can be determined by periodically examining the **TIME** column), you will want to get rid of it. Sometimes, the **kill** command will be successful in terminating a defunct process. If it is not, your only other choice is to reboot the system (which will definitely eliminate the defunct process).

Recapping the Guidelines

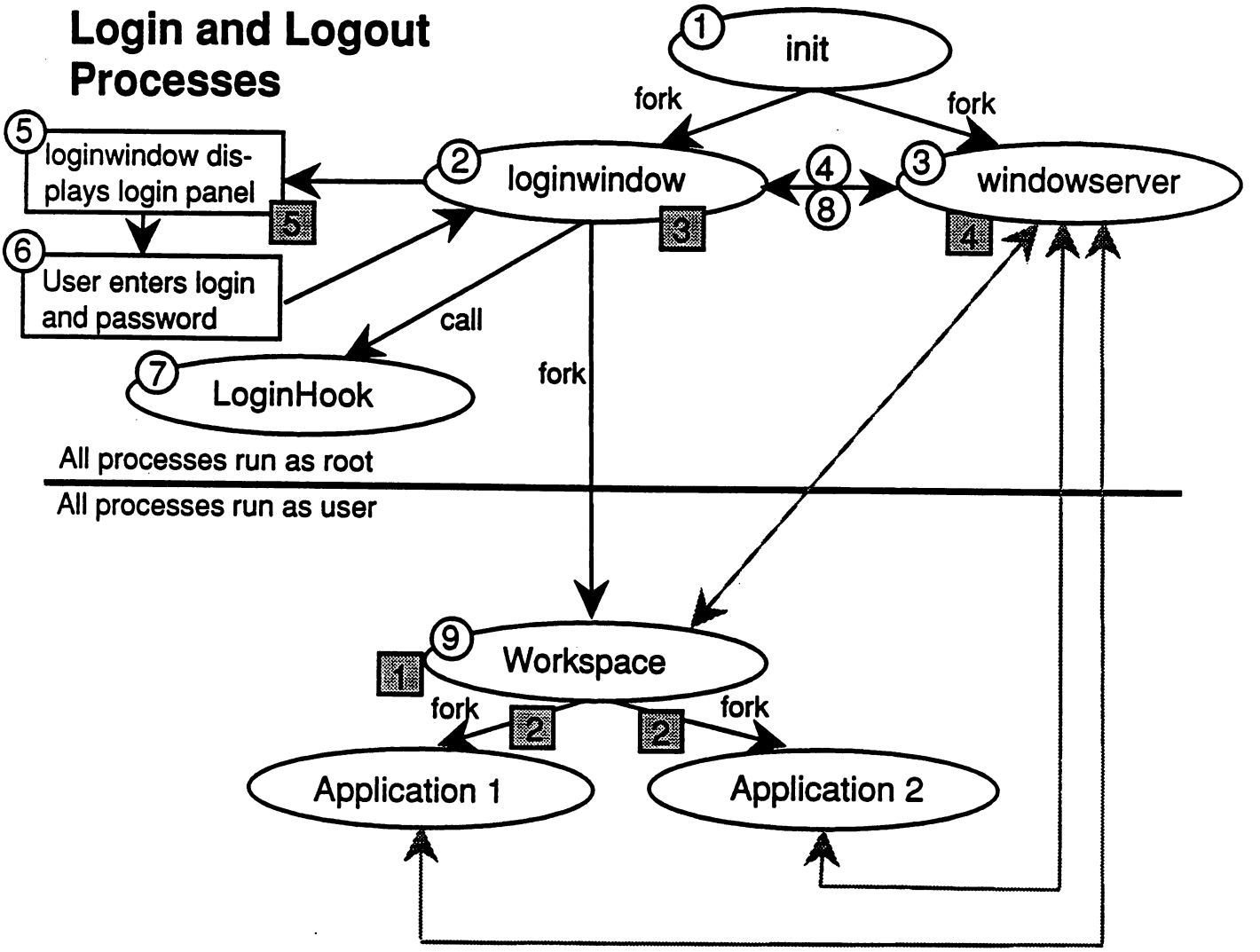
So, the rules are this:

1) Always start with a plain **`kill`** before using **`kill -9`**.
2) Always kill the children first
3) Attempt killing as a regular user and only become the super user if required.

User id and Process id



Login and Logout Processes



○ Login Sequence

1) kernel fires up init (**init -xx**)
2) init checks /etc/ttys and forks loginwindow (**-console (loginwindow)**)
3) after checking /etc/ttys, init also forks PostScript (**-console (WindowServer)**)
4) loginwindow establishes two way connection with PostScript
5) loginwindow displays login panel
6) user enters login and password
7) loginwindow optionally forks program specified as argument to LoginHook in /etc/ttys
8) loginwindow closes login panel and tells PS to change uid and gid to user's uid and gid
9) loginwindow forks Workspace Manager (**WorkSpace -LoginProgram YES**)

▩ Logout Sequence

1) user asks Workspace Manager to logout
2) Workspace Manager attempts to kill all applications and exits
3) loginwindow gets exit signal and tells PostScript to
4) close connections to all other applications and change its uid and gid back to root and daemon
5) loginwindow redisplays login panel

What's in /etc/ttys

The contents of /etc/ttys: The **/etc/ttys** file on the NeXT system is a standard 4.3 BSD **/etc/ttys** file with the addition of one new field. A typical entry in our **/etc/ttys** file looks like this:

```
console /usr/lib/NextStep/loginwindow NeXT on secure\
window=/usr/lib/NextStep/WindowServer onoption="/usr/etc/getty std.9600"
```

The fields: The first field identifies the port on which to run the process, in this case, *console* (the NeXT monitor port). The second field is the name of the program to run, (**`loginwindow`**, which allows a user to log in). The word *NeXT* identifies which type of terminal is attached to the port, the word *on* indicates that this process should be run at boot time, and *secure* means that it is ok for root to log in at this terminal. The *window* field is used to start a second process before the main process for the port, **`WindowServer`**. The last field is a NeXT addition, **onoption.**

The onoption field: The new **onoption** field in **/etc/ttys** was created for situations where the primary program (in this and most cases, **`loginwindow`**) exits with a non-zero status (indicating an error condition).

Exiting loginwindow with a non-zero exit status: **`loginwindow`** will normally only exit under one of four conditions. The two conditions listed below cause **`loginwindow`** to exit with a non-zero exit status (an error condition):

sending PostScript a kill signal
entering the word **`console`** in the login panel

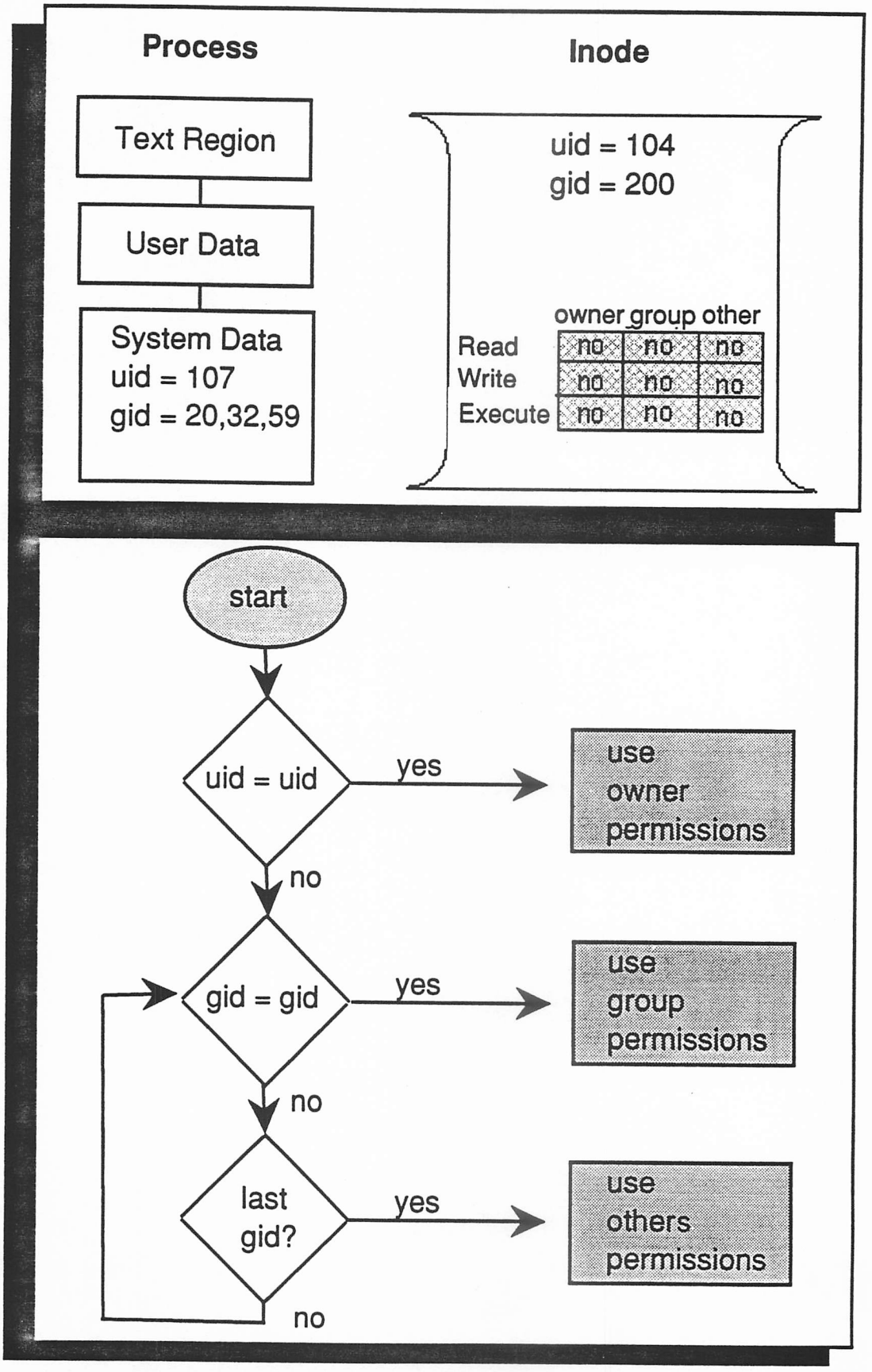
If either of these events occur, you will be presented with a login banner courtesy of **getty.** This is due to the existence of the **onerror** field in the file **/etc/ttys.** As noted above, the program listed in this field will be executed when the primary program exits with a non-zero exit status. To return to normal operations, simply log in (as anybody) and immediately log out again. You will be presented with the familiar **`loginwindow`** panel.

Exiting loginwindow with a zero exit status: loginwindow will exit with a zero exit status if either of the following two conditions arise:

sending **loginwindow** a kill signal
entering the word **exit** in the login panel

In either case, a Mach window will appear briefly, and then the login panel will be presented. These two events cause both **loginwindow** and PostScript to be restarted, and can come in handy if PostScript seems to be scrogged.

Determining Permissions



ROM Monitor Commands

The NeXT ROM monitor automatically performs self-test diagnostics and initiates the boot procedure when the system is powered up or restarted. Less frequently, the ROM monitor is used interactively to perform specialized functions. The more frequently used of these infrequently used functions, and how to perform them, are described below. For more information on the NeXT monitors see Chapter 17 in the *NeXT System Reference Manual.*

Note: All monitor and mini-monitor commands can be abbreviated. The minimum abbreviation for each command is in bold face.

Using the NMI Mini-monitor

Accessing the NMI Mini-monitor: To enter the NMI (non-maskable interrupt) mini-monitor from a running system, enter **Command-Command-~**. You will be greeted by the mini-monitor's friendly **nmi>** prompt.

Displaying the Commands that can be Executed from the NMI Mini-monitor: To display a listing of all of the commands that can be executed from the NMI mini-monitor, enter **?** at the NMI prompt. (Because of the size of the nmi window, the first several commands cannot be read.)

Returning to the Operating System from the NMI Mini-monitor: To resume execution after an escape to the NMI mini-monitor, enter **continue**.

Displaying System Messages from the NMI Mini-monitor: To display the kernel message buffer which includes the machine's Ethernet address, enter **msg**.

Shutting Down the System from the NMI Mini-monitor: If you want to shut the system down and are in contact with the NMI or panic mini-monitor, and the power switch is not responding, enter **halt**. You should also use this procedure to reboot the system if you are stuck in either mini-monitor.

Rebooting the System from the NMI Mini-monitor: To reboot the system, enter **reboot** at the NMI prompt. This will save files to disk and then reboot with the last boot command used.

Accessing the Monitor from the NMI Mini-monitor: The ROM monitor itself is not directly accessible from the operating system. To access the ROM monitor from the operating system you must first access the NMI mini-monitor and then enter monitor. Please note that this is *not* a clean shut down and should only be performed when nothing else works.

Using the ROM Monitor

Accessing the Monitor During the Boot Process: To access the ROM Monitor during the boot process, enter **Command-~** as soon as the message **testing system** is replaced by the **loading from disk** message. You will be greeted by the Monitor's **NeXT>** prompt.

Booting from the Monitor Using Default Parameters: To boot the system from the monitor using the default boot device enter **boot**.

Booting from a Non-default Device: To boot the system from a device that is not the default device enter the command listed below that corresponds to the device from which you wish to boot:

boot device	command
optical disk	**bod**
SCSI disk	**bsd**
ethernet	**ben**

The **b** in each of these commands stands for boot, **od** stands for optical disk, **sd** for SCSI disk, and **en** for Ethernet. Each can be followed by a **-s** to boot the system from the specified device in single user mode (e.g., **bod -s**).

Ejecting an Optical Disk: To eject an optical disk from the monitor, enter the command **eject**. Please see the handout titled *Ejecting an Optical Disk.*

Setting the Hardware Password: If security is an issue at your site, you will probably want to restrict access to the ROM monitor commands. This is possible by setting a hardware password. To set the password, enter **P** (that's a capital p). Once set, three configuration parameters (described in the following section) determine which monitor commands will be restricted.

Changing default boot parameters: Which device is the default boot device can be set throught the **Preferences** application or from the ROM monitor. It is also possible to change whether or not several power-on tests are performed, as well as set several security parameters. The basic method for changing any of these defaults is the same. You enter the parameter command at the ROM monitor's **NeXT>** prompt, which initiates a dialogue like the one illustrated below:

```
NeXT>p
    boot command: od?
    DRAM tests: yes?
    perform power-on system test:  yes?
                sound out tests:  yes?
                SCSI tests:  no?
                loop until key press:  no?
                verbose test mode:  no?
    boot extended diagnostics: no?
    serial port A is alternate console: no?
    allow any ROM command even if password protected: no?
    allow boot command from any device even if password protected: no?
    allow optical drive #0 eject even if password protected: yes?
```

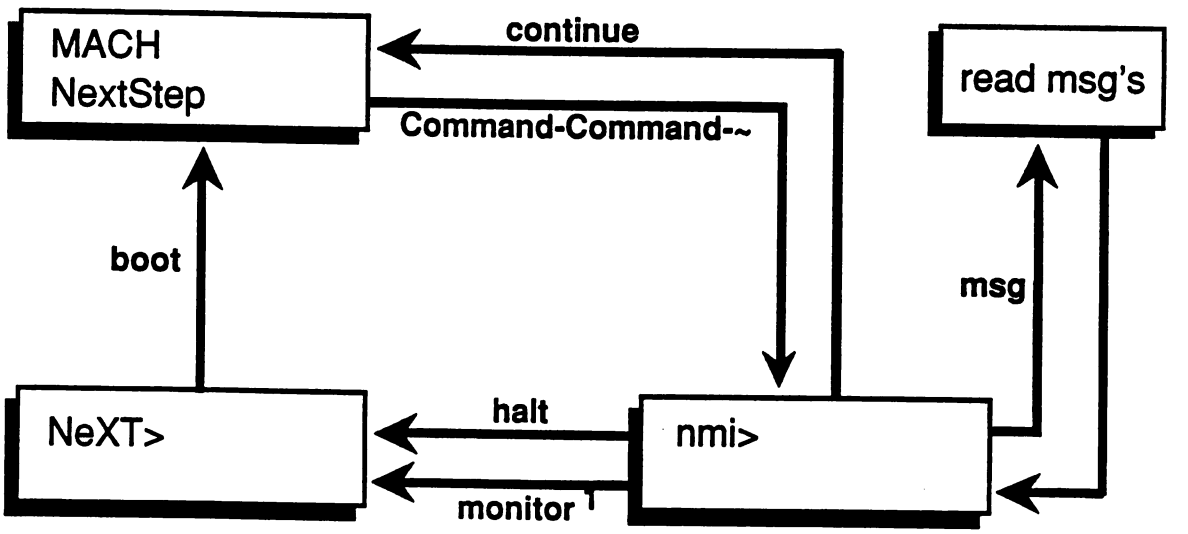
Pressing **Return** in response to any of the monitor's questions will result in the value named before the question mark remaining in effect. To change a value simply specify the new value after the question mark and press **Return**.

The possible values for the boot command are the name of the device you wish to be your default boot device as listed below (note that they are *not* preceded by a b):

boot device	command
optical disk	**od**
SCSI disk	**sd**
ethernet	**en**
null	**.**

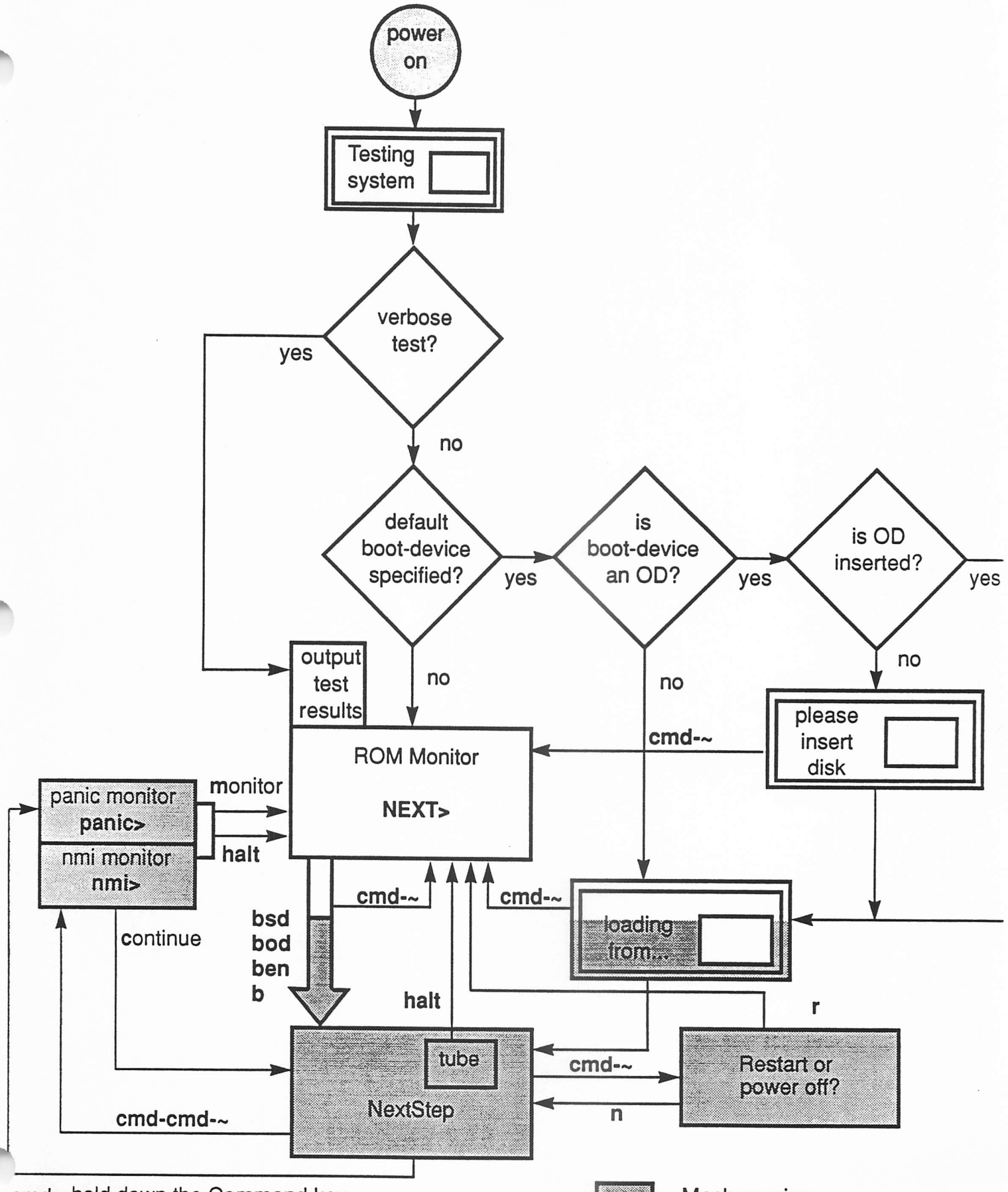
The remaining parameter values are boolean. Here you can answer **yes** or **y** to have the test performed, **no** or **n** to have the test not performed, or simply press **Return** to leave the values as they are. The final three parameters determine which ROM commands will be protected when the hardware password is set.

Monitor Command Sequence



[1] Not a clean shutdown. Use only when **halt** doesn't work.

Monitor Map



cmd= hold down the Command key

Alt = hold down the Alternate key

Alt-cmd-*(hard reset) will always return to "testing system", regardless of system state

= Mach running

Booting from a Local Disk

There are three phases to system booting:

Bootstrap—loading the Mach kernel into memory
Startup—kernel internal initialization
Init —daemon and services initialization

These are handled by the bootcode on the PROM, the kernel, and the init program respectively.

One of the design goals of the booting sequence is to provide autonomy between the three phases. Thus, the kernel needs no information about the device it was loaded from to become fully functional. The init scripts need no information from the kernel to bring the system up in multi-user mode.

Bootstrap—The ROM Monitor

The PROM bootcode can load the kernel into memory from a SCSI disk, from an optical disk, or from the network. This handout focuses on standalone startup.

To load the kernel from a local disk (SCSI or optical), the PROM bootcode reads in a "boot" program which traverses the file system on the disk, finds the kernel, and loads it into memory.

At this point, the boot program starts the kernel running.

Startup—The Mach Kernel

The Mach startup procedure involves:

- initializing hardware (since the kernel makes no assumptions about the state of the machine when it is started, it resets every hardware device)
- initializing kernel data structures
- starting kernel threads
- mounting the root file system

Some information is passed from the ROM Monitor to the kernel. This includes what the console device is and what the boot device was (SCSI, optical, or network). By default, the kernel uses the boot device for the root file system (making it the root device). This can be overridden with the **-a** boot option, which causes the kernel to prompt for the root device on the console.

If the root device is a local disk (i.e. SCSI or optical) then partition **a** is mounted as the root file system. The hostname of the system is set from the disk label of the root device.

Once the file system is set up, the kernel starts the **/etc/init** program, which controls bringing the system up in multi-user mode.

Init—The init Program

The **init** program first runs the shell script **/etc/rc.boot**. This script does enough initialization to the system to make it usable by an expert. This includes checking the file system for consistency (if needed) and initializing the network interface.

If the system is being brought up in single-user mode (with the **-s** boot flag), **init** will run an interactive shell (**/bin/sh**) on the console. When this shell exits (or if you are not starting up in single-user mode) **init** will run the shell on the script **/etc/rc**.

/etc/rc performs some general system start-up operations and starts the system daemons then calls **/etc/rc.local**. This script is provided for you to use to specify local startup procedures.

The rc scripts have been designed to work on all system configurations. They operate in conjunction with the file **/etc/hostconfig** which is usually modifed by **NetManager**. Typically the only startup files you will modify are **/etc/hostconfig** and **/etc/rc.local**.

Startup Scripts

/etc/rc.boot Always run by **init** (unless you boot with the "-b" option)

- Prints date on console
- Checks filesystem(s) with fsck
- Sources **/etc/hostconfig**
- Fakes root mount entries
- Configures ethernet
- Sets hostname

/etc/rc Run by **init** before going into Multi-User mode

- Prints date on console
- Runs **rc.swap**
- Mounts local filesystems
- If necessary, recovers password file
- Cleans up temporary files
- Starts early daemons
- Sources **/etc/hostconfig**
- Starts other daemons (runs **rc.local**)

/etc/rc.swap Run by **/etc/rc** to mount and page to an alternate swap disk.

/etc/rc.local Run by **/etc/rc** to start locally installed (non-NeXT) software.

Daemon Processes

Daemon	Function
kernel-task	the kernel process
init	the first true process
mach-init	controls system service port inheritance
kern_loader	loadable device driver server
syslogd	appends system error messages to logfile
portmap	converts RPC program numbers into DARPA protocol numbers
nibindd	daemon that start NetInfo service; spawns netinfod daemons
netinfod	net info server (one for each domain served on machine)
routed	network daemon for routing packets over a gateway
biod 4	asynchronous block I/O daemons—provides caching for network filesystems
nmserver	supports networking of Mach Inter-Process Communication and acts as name server for Mach IPC ports
inetd	initial InterNet dameon
sendmail	mail sending daemon
lpd	line printer daemon
pbs	NeXT Paste Board Server
autodiskmount	automounter daemon
npd	daemon that intercedes between applications and lpd, determining whether to send PostScript directly to the Window Server or to be spooled
np_buddy	daemon created by printer device driver for print jobs
update	runs **sync** every 30 seconds
cron	executes automated system processes at specified times
telnetd	telnet server
nfsd 4	daemons that handle client filesystem requests
loginwindow	responsible for controlling windowserver and workspace

User Information

Creating a UNIX user account involves storing several pieces of information about the user in a database. The traditionl UNIX ASCII file for this database is **/etc/passwd**, which contains one record of colon-separated fields per line. On a NeXT system, this information is stored in the **NetInfo** data files. When the system needs to reference this information, the **NetInfo** data is checked first, then the ASCII file. If the ASCII file references **yp**, then the **yp** data files are checked. The fields included in the database are:

User (login) Name: Each user on the system must have a unique login name, consisting of no more than eight characters. User names, by convention, are all lower case letters (remember, UNIX is case sensitive).

Password: Each user should be assigned a password. When stored in the database, the password is encrypted for security reasons.

User Identification Number (UID): Each account must be assigned a unique identification number. This number is actually used by the system to store with files and processes, and the user names are purely for human convenience (it's a lot easier to remember several people's login names than a lot of user id numbers).

Group Identification Number (GID): Each user must belong to at least one user group (more below). The firt such group is identified here, by number (GID). Additional group memberships are specified in the groups data file.

Other: Frequently, a text field is available for storing miscellaneous information about a user, usually the full name of the person associated with the user account. Utilities such as **`finger`** exist that access this field, although it is frequently left blank.

Home Directory: When a user logs in, the current working directory will be whatever directory is listed in this field (the directory must exist and have the appropriate ownerships and permissions set for the user *before* they log in for the first time).

Login Shell: On traditional UNIX systems, this would be the program that is run when the user logs in. On NeXT systems, this is the name of the program that is run when either the **Terminal** or **Shell** application is launched. In most cases, this will be **/bin/csh** (the C-Shell).

Group Information

Information about user groups is also stored in a data file. As with the password data, the system uses one of three files for the information: the **NetInfo** database, the **yp** database, or the ASCII file **/etc/group**. The fields stored for groups are:

Group Name: Just as with user accounts, each user group should have its own unique name.

Group Id Number (GID): Again, as with a user account, each group must have a unique number asssigned. This number is what is used by the system to identify the group.

Members: A list of the user names belonging to the group.

Group Password: On some systems, the capability exists to assign a password for each group. This is rarely used, but space may still be provided for this field in the group database.

Housekeeping Files

Home directories are created with several standard housekeeping files. In addition, many UNIX utilities and NeXT applications use the fact that files with names that begin with a "." are not displayed by default, to hide their housekeeping files from novice users. All of the files discussed here can be found in or below users' home directories. The names of these files (relative to a user's home directory) and their corresponding functions are listed below.

Note: Files or directoies with names that begin with a "." will not appear in the Browser unless you select **Filter...** from the **View** sub-menu of the **Workspace Manager** and then click the **Special Files** button.

NeXT Housekeeping Files Found in the .NeXT Directory

Name	Function
.NeXTtrash	This directory contains any files or directories that have been dragged into the black hole and have not been destroyed.
.dock	This non-ASCII file contains a list of all docked applications and is updated as icons are added or removed from the dock. The **Workspace Manager** reads this file at start-up and places all listed icons on the dock.
.NeXTdefaults.L	This non-ASCII file contains a list of default parameters.
.NeXTdefaults.D	This non-ASCII file is an index for **.NeXTdefaults.L.**
targets	This directory is used by the **Digital Librarian** to track the contents of its bookshelf.

UNIX Housekeeping Files Found in the User's Home Directory

Name	Function
.cshrc	This file is read by the C-shell when a user who has the C-shell specified as their default shell spawns a new shell. This usually occurs when a user logs in over the network or launches a **Shell** or **Terminal** (this file is read before **.login**).
.login	This file is read by the C-shell when a user who has the C-shell specified as their default shell logs in over the network or launches a **Shell** or **Terminal** (this file is read after **.cshrc**).
.profile	This file is read by the Bourne shell when a user who has the Bourne shell specified as their default shell logs in over the network or launches a **Shell** or **Terminal.**
.mailrc	This file is read by the generic UNIX **`mail`** program when a user invokes the UNIX mail utility.
.plan	This file is read by the UNIX **`finger`** program.
.project	This file is read by the UNIX **`finger`** program.

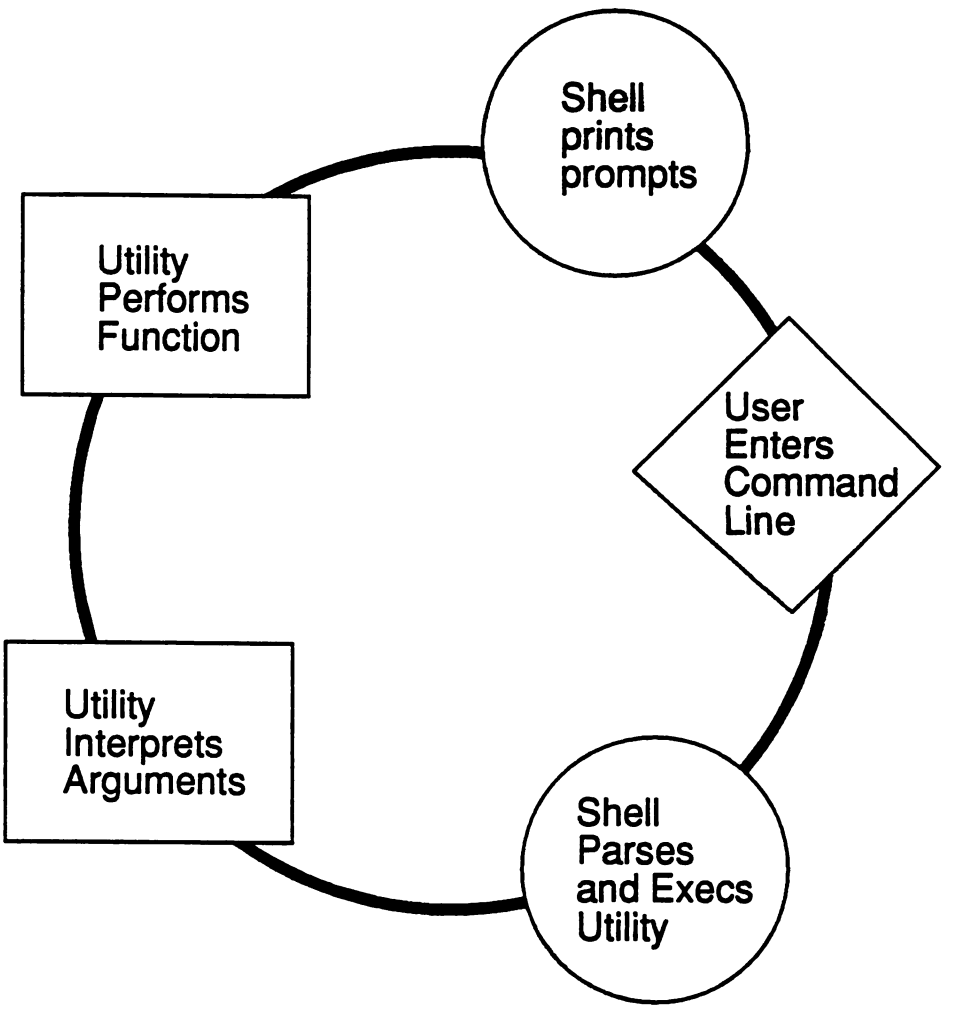
Other Dot Files

Name	Function
.places	This file is used by the **Workspace Manager** to keep track of a directory's contents and to record icon positions, view type, window size.
.list	This file is used by the **Edit** Application to keep track of a directory's contents.
/.hidden	A list of files to be shown only to UNIX experts.

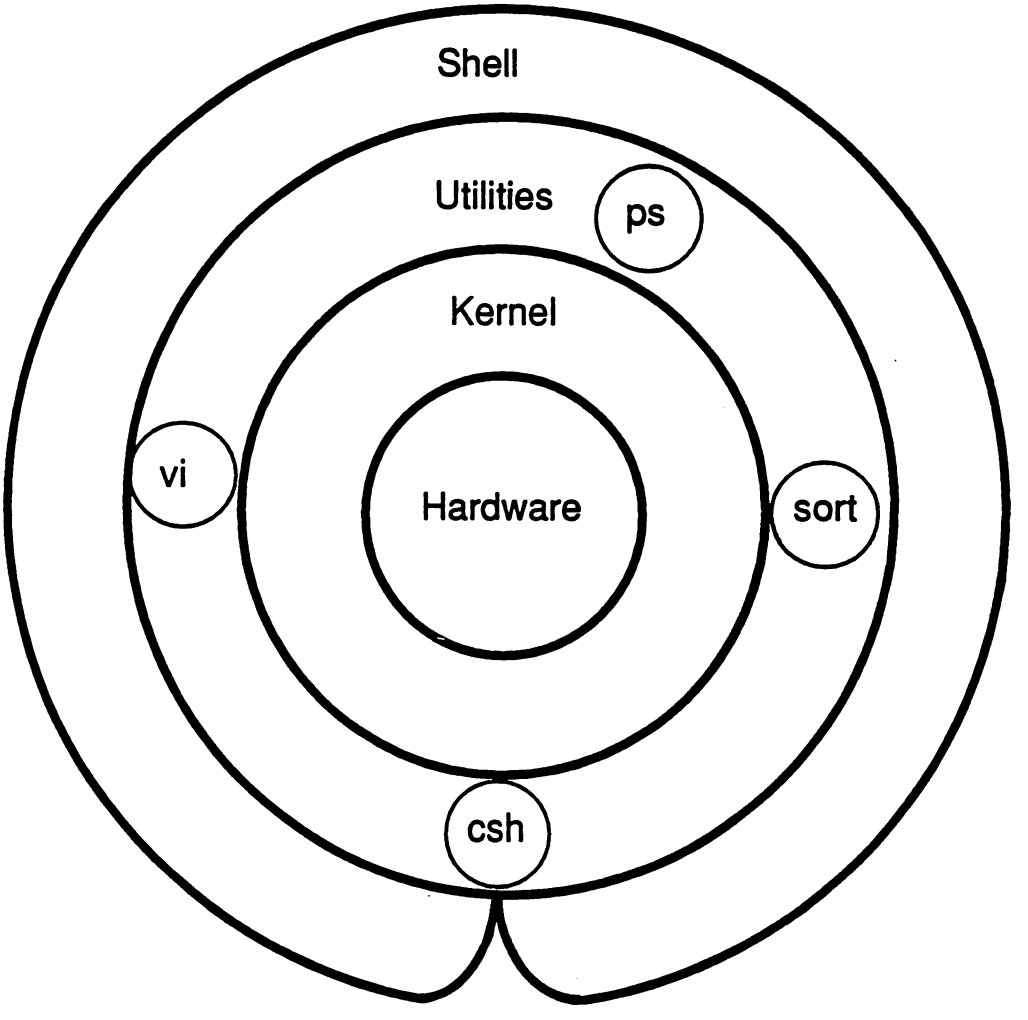
The Problems With Current Desk Top Software

- Command line interfaces are easy to program and hard to use
- Graphic interfaces are easy to use and hard to program
- Standardized graphic interfaces are easier to use and harder to program
- People are multitasking and want their computers to do the same
- Multiple imaging models
- Networking is an afterthought

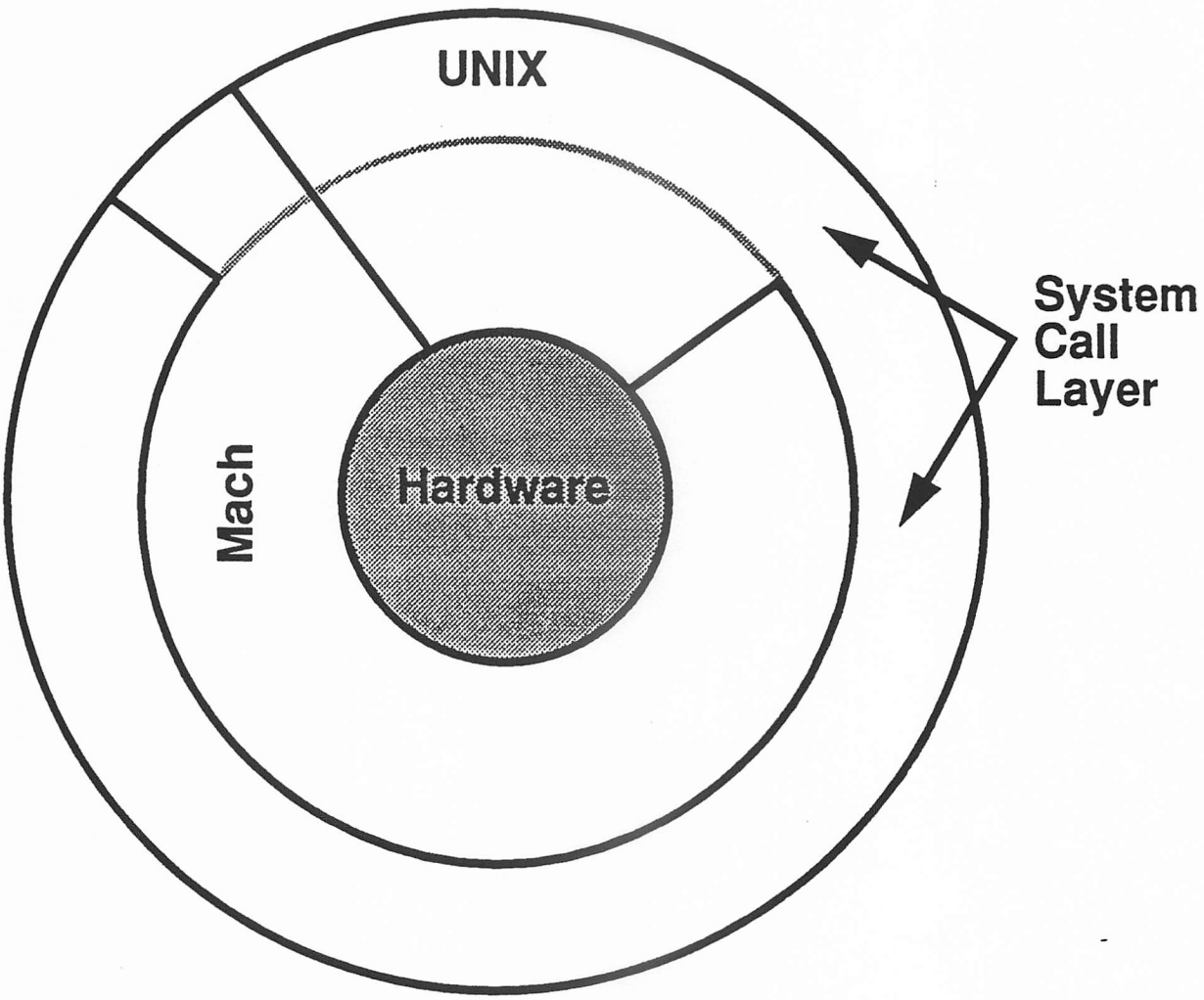
Shell Sequence of Interpretation



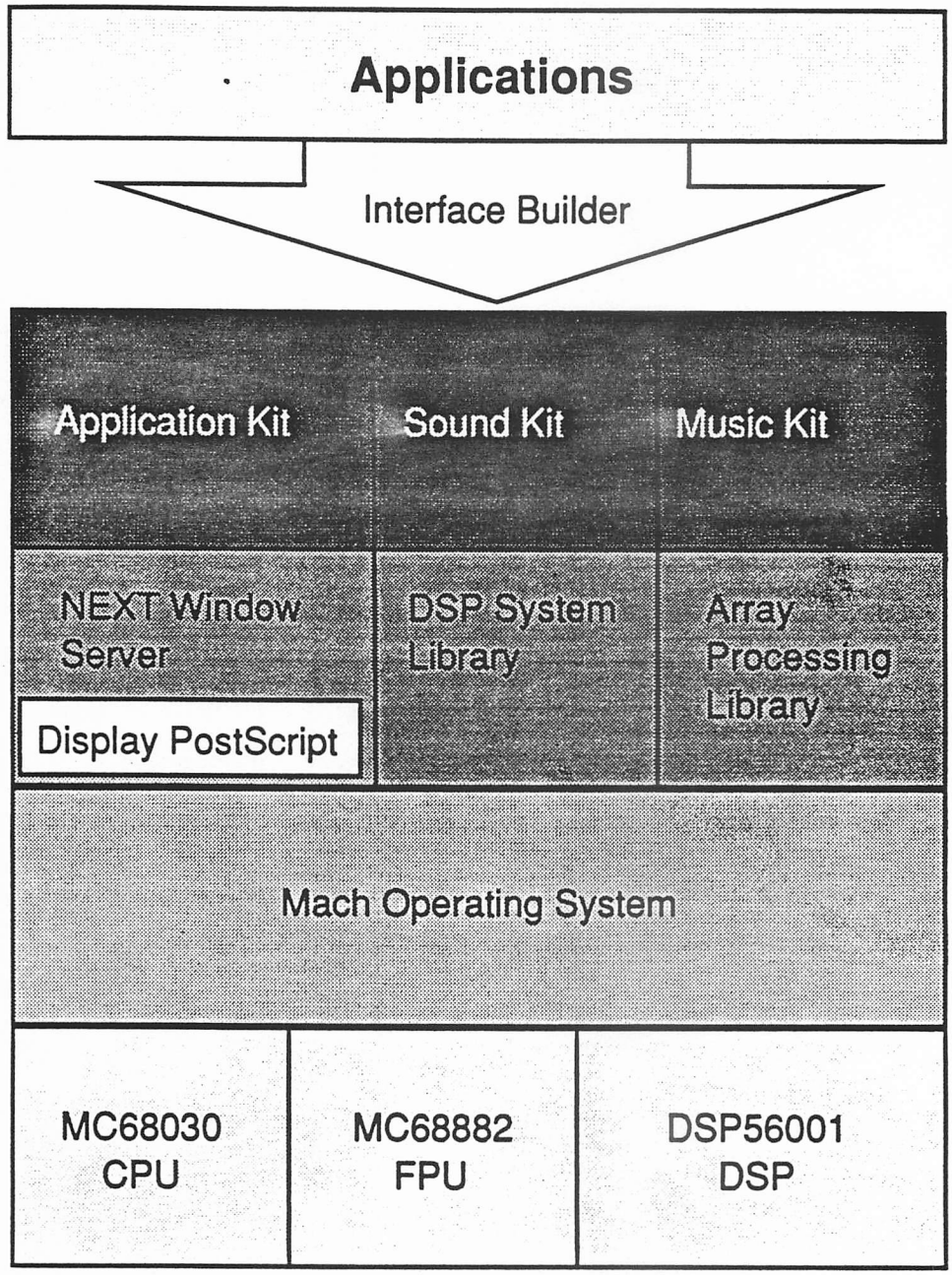
Modular Components of the UNIX Operating Environment



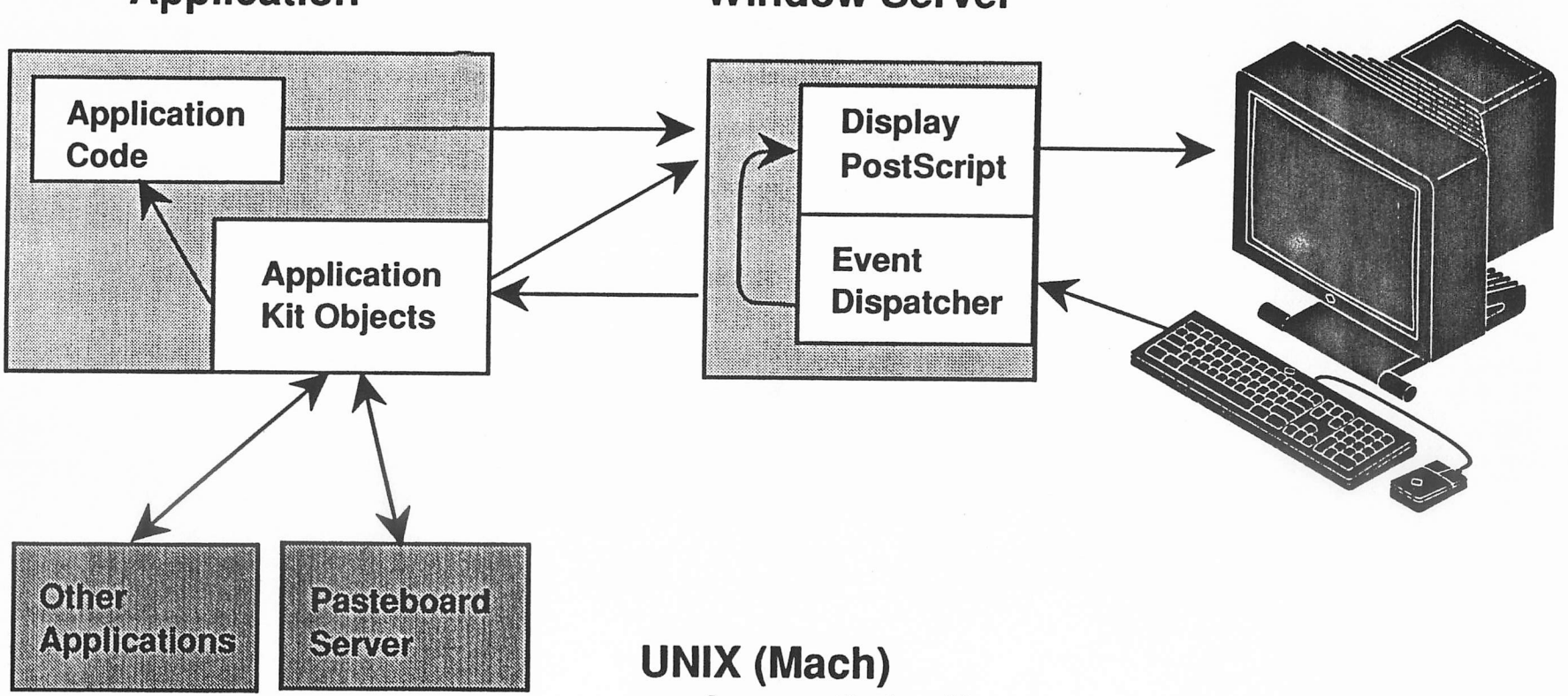
Mach and UNIX



System Overview



NeXT Software

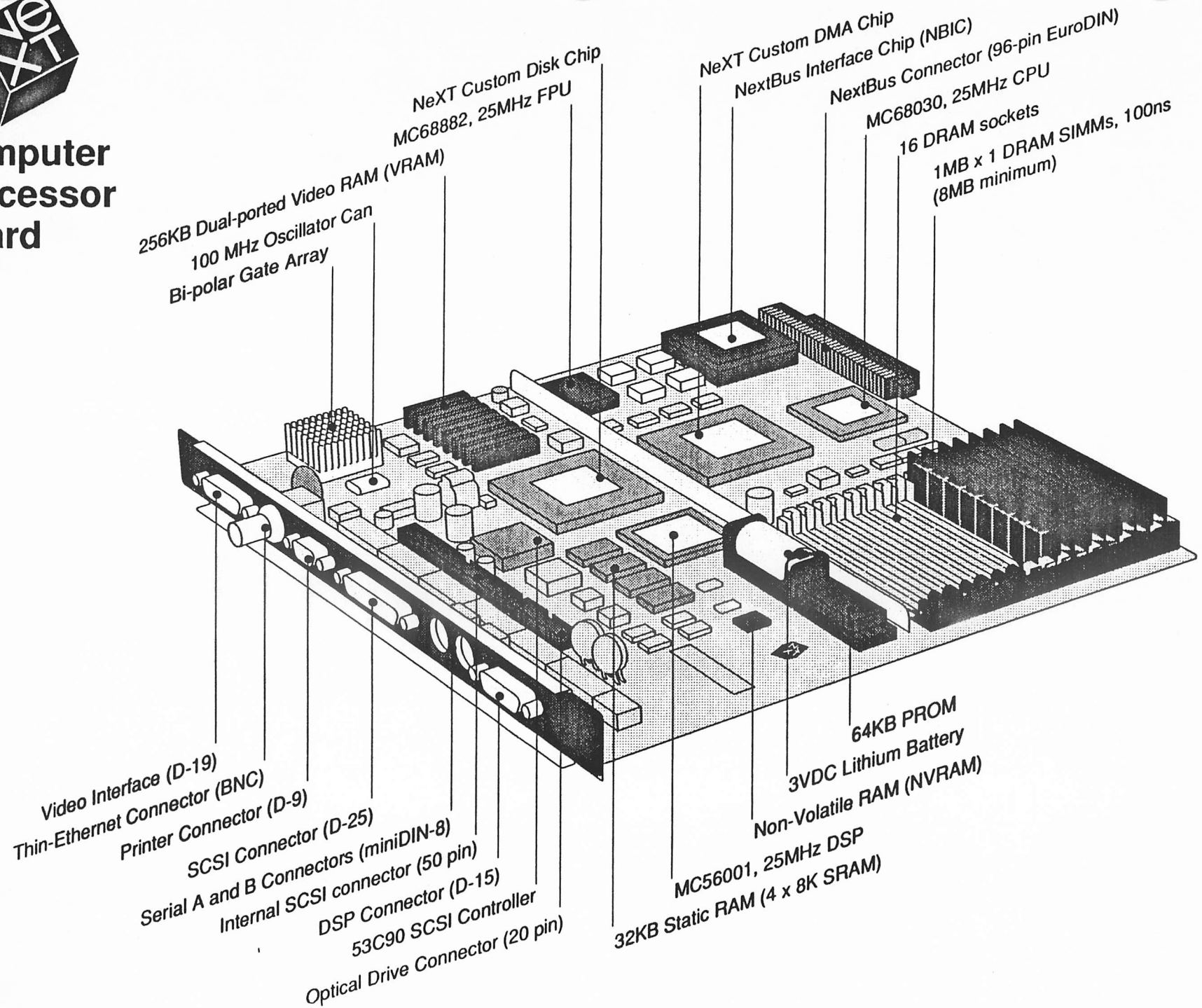


UNIX (Mach) underneath it all

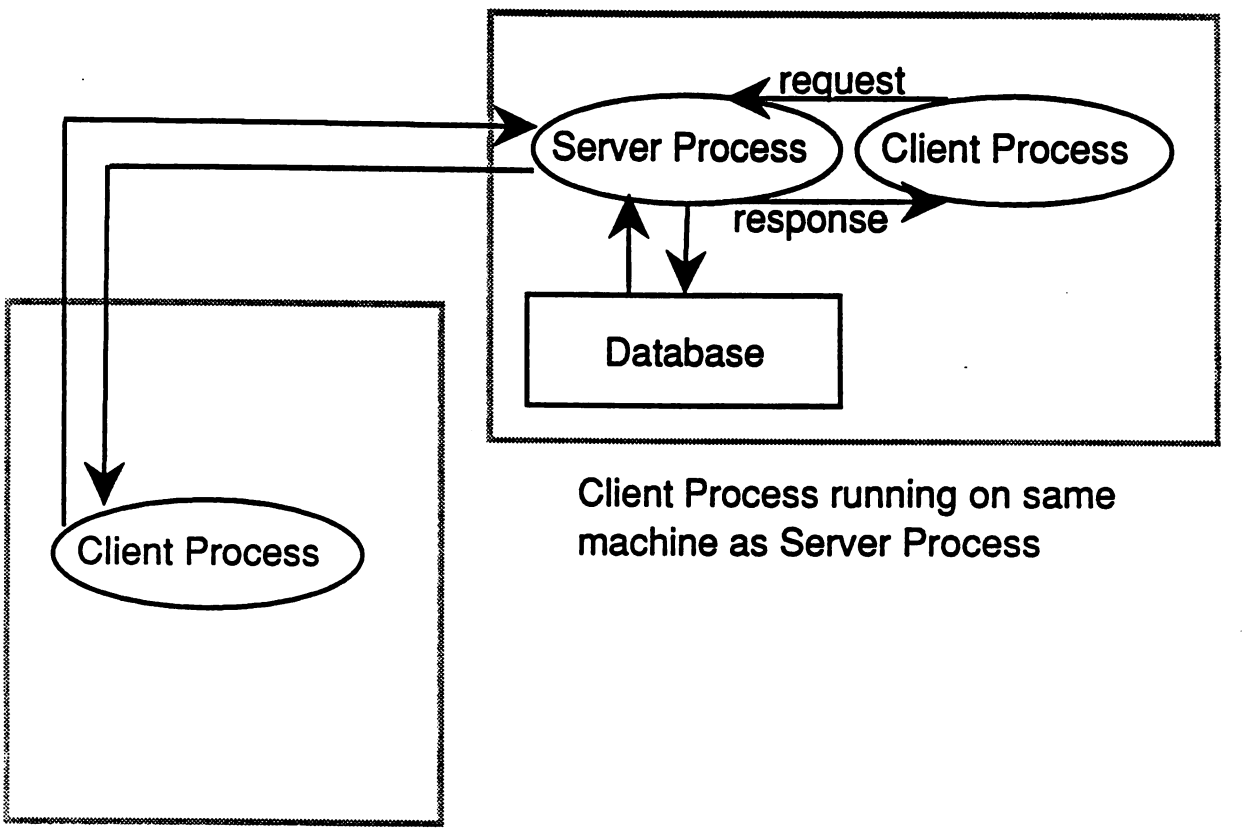
Maximizing Software Efficiency

- Mach virtual memory
- "Copy-on-write" memory
- Shared libraries
- Mach IPC
- Tasks and threads

Computer Processor Board



Client and Server Process Interaction



Client Process running on same machine as Server Process

Client Process running on a separate machine

NAME

ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

/usr/etc/ping [**–r**] [**–v**] *host* [*packetsize*] [*count*]

DESCRIPTION

The DARPA Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking a single-point hardware or software failure can often be difficult. *Ping* utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a **struct timeval**, and then an arbitrary number of "pad" bytes used to fill out the packet. Default datagram length is 64 bytes, but this may be changed using the command-line option. Other options are:

–r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by *routed*(8C)).

–v Verbose output. ICMP packets other than ECHO RESPONSE that are received are listed.

When using *ping* for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be "pinged". *Ping* sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. No output is produced if there is no response. If an optional *count* is given, only that number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times out (with a *count* specified), or if the program is terminated with a SIGINT, a brief summary is displayed.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use *ping* during normal operations or from automated scripts.

AUTHOR

Mike Muuss

SEE ALSO

netstat(1), ifconfig(8C)

NAME

telnet – user interface to the TELNET protocol

SYNOPSIS

telnet [host [port]]

DESCRIPTION

Telnet is used to communicate with another host using the **TELNET** protocol. If *telnet* is invoked without arguments, it enters command mode, indicated by its prompt ("telnet>"). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command (see below) with those arguments.

Once a connection has been opened, *telnet* enters an input mode. The input mode entered will be either "character at a time" or "line by line" depending on what the remote system supports.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially "^E") may be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

In either mode, if the *localchars* toggle is TRUE (the default in line mode; see below), the user's *quit*, *intr*, and *flush* characters are trapped locally, and sent as **TELNET** protocol sequences to the remote side. There are options (see **toggle** *autoflush* and **toggle** *autosynch* below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the **TELNET** sequence) and flush previous terminal input (in the case of *quit* and *intr*).

While connected to a remote host, *telnet* command mode may be entered by typing the *telnet* "escape character" (initially "^]"). When in command mode, the normal terminal editing conventions are available.

COMMANDS

The following commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the **mode**, **set**, **toggle**, and **display** commands).

open *host* [*port*]

Open a connection to the named host. If no port number is specified, *telnet* will attempt to contact a **TELNET** server at the default port. The host specification may be either a host name (see *hosts*(5)) or an Internet address specified in the "dot notation" (see *inet*(3N)).

close

Close a **TELNET** session and return to command mode.

quit

Close any open **TELNET** session and exit *telnet*. An end of file (in command mode) will also close a session and exit.

z

Suspend *telnet*. This command only works when the user is using the *csh*(1).

mode *type*

Type is either *line* (for "line by line" mode) or *character* (for "character at a time" mode). The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

status

Show the current status of *telnet*. This includes the peer one is connected to, as well as the current mode.

display [*argument...*]

Displays all, or some, of the **set** and **toggle** values (see below).

? [*command*]

Get help. With no arguments, *telnet* prints a help summary. If a command is specified, *telnet* will print the help information for just that command.

send *arguments*

Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

escape

Sends the current *telnet* escape character (initially "^]").

synch

Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2 BSD system -- if it doesn't work, a lower case "r" may be echoed on the terminal).

brk

Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ip

Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

ao

Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.

ayt

Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

ec

Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el

Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

ga

Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

nop

Sends the TELNET NOP (No OPeration) sequence.

?

Prints out help information for the **send** command.

set *argument value*

Set any one of a number of *telnet* variables to a specific value. The special value "off" turns off the function associated with the variable. The values of variables may be interrogated with the **display** command. The variables which may be specified are:

echo

This is the value (initially "^E") which, when in "line by line" mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

escape

This is the *telnet* escape character (initially "^[") which causes entry into *telnet* command mode (when connected to a remote system).

interrupt

If *telnet* is in *localchars* mode (see **toggle** *localchars* below) and the *interrupt* character is typed, a **TELNET IP** sequence (see **send** *ip* above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

quit

If *telnet* is in *localchars* mode (see **toggle** *localchars* below) and the *quit* character is typed, a **TELNET BRK** sequence (see **send** *brk* above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

flushoutput

If *telnet* is in *localchars* mode (see **toggle** *localchars* below) and the *flushoutput* character is typed, a **TELNET AO** sequence (see **send** *ao* above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's **flush** character.

erase

If *telnet* is in *localchars* mode (see **toggle** *localchars* below), **and** if *telnet* is operating in "character at a time" mode, then when this character is typed, a **TELNET EC** sequence (see **send** *ec* above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

kill

If *telnet* is in *localchars* mode (see **toggle** *localchars* below), **and** if *telnet* is operating in "character at a time" mode, then when this character is typed, a **TELNET EL** sequence (see **send** *el* above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

eof

If *telnet* is operating in "line by line" mode, entering this character as the first character on a line will cause this character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's **eof** character.

toggle *arguments...*

Toggle (between TRUE and FALSE) various flags that control how *telnet* responds to events. More than one argument may be specified. The state of these flags may be interrogated with the **display** command. Valid arguments are:

localchars

If this is TRUE, then the *flush*, *interrupt*, *quit*, *erase*, and *kill* characters (see set above) are recognized locally, and transformed into (hopefully) appropriate **TELNET** control sequences (respectively *ao*, *ip*, *brk*, *ec*, and *el*; see **send** above). The initial value for this toggle is TRUE in "line by line" mode, and FALSE in "character at a time" mode.

autoflush

If *autoflush* and *localchars* are both TRUE, then when the *ao*, *intr*, or *quit* characters are recognized (and transformed into **TELNET** sequences; see **set** above for details), *telnet* refuses to display any data on the user's terminal until the remote system acknowledges (via a **TELNET** *Timing Mark* option) that it has processed those **TELNET** sequences. The initial value for this toggle is TRUE if the terminal user had not done an "stty noflsh", otherwise FALSE (see *stty(1)*).

autosynch

If *autosynch* and *localchars* are both TRUE, then when either the *intr* or *quit*

characters is typed (see set above for descriptions of the *intr* and *quit* characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure should cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

crmod

Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug

Toggles socket level debugging (useful only to the *super*user). The initial value for this toggle is FALSE.

options

Toggles the display of some internal *telnet* protocol processing (having to do with TELNET options). The initial value for this toggle is FALSE.

netdata

Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

?

Displays the legal toggle commands.

BUGS

There is no adequate way for dealing with flow control.

On some remote systems, echo has to be turned off manually when in "line by line" mode.

There is enough settable state to justify a *.telnetrc* file.

No capability for a *.telnetrc* file is provided.

In "line by line" mode, the terminal's *eof* character is only recognized (and sent to the remote system) when it is the first character on a line.

The NetInfo Suite of Manager Applications

Included with the bundled software on the NeXT computer are five applications used to manage NetInfo information. What follows is a brief description of each, to assist you in determining what application to use.

MailManager: This application, located under **/NextAdmin**, is used to set up Mail on a NetInfo network.

NetManager: This application is used to configure a NetInfo configuration server, which is part of the initial setup of a NeXT network. This application is also in **/NextAdmin.**

UserManager: **UserManager** is a tool used to add, modify and delete user accounts. It lives in **/NextAdmin.**

PrinterManager: This application is where you would go to add a new printer to your system, configure a network printer and monitor the progress of print jobs. This application, unlike the others, is located in **/NextApps.**

NetInfoManager: This is an application that allows you to access and manipulate all of the NetInfo data. Through this manager, you can perform any function not listed covered by the other manager applications (delete a user's password, add or remove a user from a group, add a new user group, etc.). **NetInfoManager** can also be found in **/NextAdmin.**

Monitoring Disk Usage

Monitoring disk usage on any computer is a necessary Support function. Users aren't really concerned with the amount of space they consume, nor are they truly aware of the limitations of the disk space available to them. A full filesystem can cause serious problems in the user community and unwanted visibility for the Support Staff. This problem has worsened as more systems have implemented servers and distributed storage devices.

The Disk Info menu command: To check the amount of free disk space on a filesystem, use the Browser to select a file within that filesystem and click the **Disk Info** command located on the Workspace Manager's **Files** menu.

The df shell command: To check the amount of free disk space on all currently mounted filesystems, launch a **Shell** and use the **df** command. The screen will display a list of the filesystems by name, their individual sizes, the amount of used and available space in each filesystem, the level of capacity, and where it's mounted. A short sample of such a listing:

```
Goofy2: df directoryname
   Filesystem          kbytes    used      avail    capacity     Mounted on
   /dev/sd0a           345711    202712    108427     65%        /
   batcomputer/mail    620091     47738    510343     09%        /LocalLibrary/Image
   k9:/dps             620091    518610     19471     97%        /Net/k9/dps
```

This display gives you an overview of your filesystem and allows easy identification of areas of possible problems. The limitation of this display is obvious: there is not enough detail to allow you to act on specific files or users to prevent a completely full filesystem from occurring. That specific detail is necessary, and easily discovered.

The du shell command: Further digging around to find the exact filesystem or files that are causing the space problem can begin with the **du** shell command. Invoking this command will generate a listing of all directories and files within the directory you specify, and the number of kilobytes used in each.

In large filesystems this is not very practical because of the length of the list that would be generated (executing in the root directory would be a lenghty unhappiness). Instead, in large filesystems, use **du -s** and all that is displayed on the screen is the total number of kilobytes used in the filesystem specified.

The following is an example of both of these commands, and the kind of ouptut they generate:

```
Goofy2: du directoryname          differs from    Goofy2: du -s directoryname
   16   ./.NeXT/.NeXTtrash                              498
   21   ./.NeXT/targets
   45   ./.NeXT
   129  ./Mailboxes/May89
   82   ./Mailboxes/Classes
   330  ./Mailboxes
   78   ./Materials
   498  .
```

Preventative Maintenance: Part of your daily routine (more or less often depending on the level of activity on your system and the amount of disk storage available) will include checking on the available space in the various filesystems in your care. The **df** and **du** commands are two of the most useful tools at the disposal of the Support Staff. Both commands can be executed as superuser in homedirectories, or wherever more information about space utilization is necessary. Their frequent and judicious use will help you avert the problems caused by a full filesystem, thus giving your user community one less thing to complain about.

Hint: Some likely places to check for the causes of space problems include:

users' home directories
.NeXT/.NeXTtrash
/me
/dev
/usr/adm (check the various logfiles for size and/or unusal growth patterns)

NAME

df – report free disk space on file systems

SYNOPSIS

df [**–i**] [filesystem ...] [file ...] [**–t** *type*] [*filesystem* ...] [*filename* ...]

DESCRIPTION

df displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. Used without arguments, *df* reports on all mounted file systems, producing something like:

```
tutorial% df
Filesystem  kbytes  used  avail  capacity  Mounted on
/dev/ip0a    7445   4714  1986   70%       /
/dev/ip0g   42277  35291  2758   93%       /usr
```

Note that used+avail is less than the amount of space in the file system (kbytes); this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this may be adjusted using *tunefs*. When all the space on a file system except for this reserve is in use, only the super-user can allocate new files and data blocks to existing files. When a file system is overallocated in this way, *df* may report that the file system is more than 100% utilized.

If arguments to *df* are disk partitions (for example, */dev/ip0a*) or UNIX path names, *df* produces a report on the file system containing the named file. Thus "df ." shows the amount of space on the file system containing the current directory.

OPTIONS

–i Also report the number of inodes which are used and free. This option is only meaningful for local file systems, since the NFS protocol does not support it.

–t *type* Report on filesystems of a given *type* (for example, nfs or 4.2).

FILES

/etc/mtab list of currently mounted filesystems

SEE ALSO

du(1V), mtab(5), quot(8), tunefs(8)

NAME

du – summarize disk usage

SYNOPSIS

du [**–s**] [**–a**] [name ...]

DESCRIPTION

Du gives the number of kilobytes contained in all files and, recursively, directories within each specified directory or file *name*. If *name* is missing, '.' is used.

The argument **–s** causes only the grand total to be given. The argument **–a** causes an entry to be generated for each file. Absence of either causes an entry to be generated for each directory only.

A file which has two links to it is only counted once.

SEE ALSO

df(1), quot(8)

BUGS

Non-directories given as arguments (not under **–a** option) are not listed.
If there are too many distinct linked files, *du* counts the excess files multiply.

Ejecting an Optical Disk

There are four methods that can be used to eject an optical disk. The first is for most situations, while the second is mostly available as an added convenience. The third and fourth methods are for use in unusual circumstances.

Ejecting an Optical Disk from the WorkSpace Manager: In most situations, you will use the Browser to highlight a file or directory that is stored on the OD. Once highlighted, selecting **Unmount** from the **Optical** submenu will eject the OD and unmount it from the directory structure. Selecting **Mount** will eject the current disk without unmounting and prompt for a new disk to be inserted.

Ejecting an Optical Disk with the disk Command: To eject an optical disk with the **disk** command, you must first launch a Shell and become the super user with the **su** command.

If the disk is mounted, you must first unmount it:

```
umount  /dev/od0a
```

To eject the disk, enter the command:

```
disk -e  /dev/rod0a
```

The **-e** specifies the eject action to the **disk** utility.

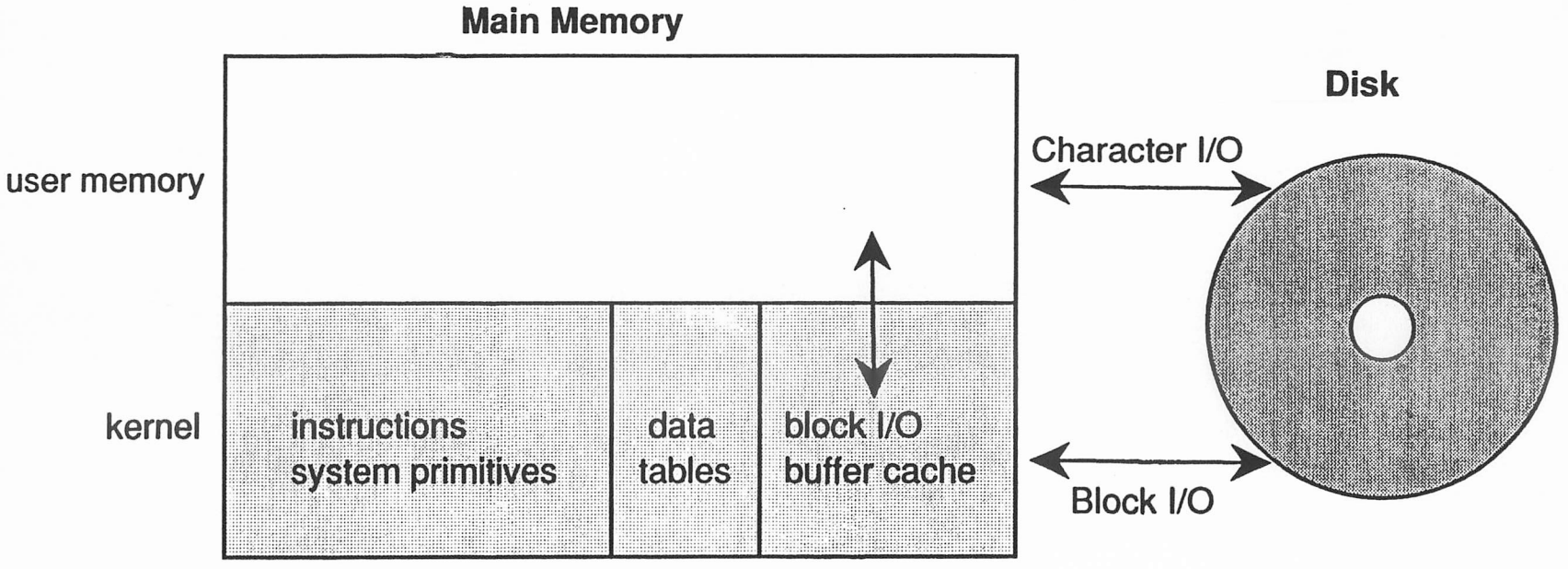
Ejecting an Optical Disk from the ROM Monitor: From the ROM Monitor's **NeXT>** prompt, you can enter the command **ej** to eject your optical disk.

Manually Ejecting a Disk: If all else fails, the only thing left is the brute force approach.

Insert your magic tool in the small slot above the right side of the optical disk insertion slot and push in until the disk ejects.

This will leave the disk loading mechanism in a locked state. Fear not, pushing the system through its power cycle (i.e. turning it off and then on again) will fix the problem.

Overall View of Memory



NAME

fsck – file system consistency check and interactive repair

SYNOPSIS

/usr/etc/fsck –p [filesystem ...]
/usr/etc/fsck [**–P**] [**–b** block#] [**–y**] [**–n**] [filesystem] ...

DESCRIPTION

The first form of *fsck* preens a standard set of filesystems or the specified file systems. It is normally used in the script /etc/rc during automatic reboot. In this case *fsck* reads the table **/etc/fstab** to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible. Normally, the root file system will be checked on pass 1, other "root" ("a" partition) file systems on pass 2, other small file systems on separate passes (e.g. the "d" file systems on pass 3 and the "e" file systems on pass 4), and finally the large user file systems on the last pass, e.g. pass 5. Only partitions in fstab that are mounted "rw" or "rq" and that have non-zero pass number are checked.

The system takes care that only a restricted class of innocuous inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

Unreferenced inodes

Link counts in inodes too large

Missing blocks in the free list

Blocks in the free list also in files

Counts in the super-block wrong

These are the only inconsistencies that *fsck* with the **–p** option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature of the correction. After successfully correcting a file system, *fsck* will print the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

If sent a QUIT signal, *fsck* will finish the file system checks, then exit with an abnormal return status that causes the automatic reboot to fail. This is useful when you wish to finish the file system checks, but do not want the machine to come up multiuser.

Without the **–p** option, *fsck* audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that some of the corrective actions which are not correctable under the **–p** option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond **yes** or **no**. If the operator does not have write permission on the file system *fsck* will default to a **–n** action.

Fsck has more consistency checks than its predecessors *check*, *dcheck*, *fcheck*, and *icheck* combined.

The following flags are interpreted by *fsck*.

–P This flag forces *fsck* to check the disk even if the clean flag is set on the disk's superblock.

–b Use the block specified immediately after the flag as the super block for the file system. Block 16 is always an alternate super block.

–y Assume a yes response to all questions asked by *fsck;* this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

–n Assume a no response to all questions asked by *fsck;* do not open the file system for writing.

If no filesystems are given to *fsck* then a default list of file systems is read from the file **/etc/fstab**.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   Directory size not of proper format.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   File pointing to unallocated inode.
   Inode number out of range.
8. Super Block checks:

   More blocks for inodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. If the *lost+found* directory does not exist, it is created. If there is insufficient space its size is increased.

Checking the raw device is almost always faster.

FILES

/etc/fstab contains default list of file systems to check.

DIAGNOSTICS

The diagnostics produced by *fsck* are fully enumerated and explained in Appendix A of "Fsck – The UNIX File System Check Program" (SMM:5).

SEE ALSO

fstab(5), fs(5), newfs(8), mkfs(8), crash(8V), reboot(8)

BUGS

There should be some way to start a **fsck –p** at pass *n*.

Booting from a Network

Booting from a network is done in the same three phases as booting from a local disk.

Bootstrap—loading the Mach kernel into memory
Startup—kernel internal initialization
Init—daemon and services initialization

As in local booting, these phases are handled by the PROM boot code, the kernel, and the init program respectively. The init phase is identical to booting from a local disk. Thus, this document summarizes the additional PROM and kernel steps required to boot from a network.

Bootstrap—The PROM Boot Code

To boot from a network the ROM boot code must load the kernel into memory from the network. This requires the following four steps:

- The PROM first broadcasts a BOOTP request (RFC951) containing its Ethernet address.

- All net–boot servers on the net then use their own **bootpd** to consult their local **bootptab** file and the **NetInfo** database and then return the requestor's Internet (IP) Address and the path to the Mach kernel on the server (see the *UNIX Programmers Manual* entry for **bootpd**).

- The PROM then configures its Ethernet interface to use the IP address returned in the BOOTP response.

- Next, the PROM downloads the kernel into memory from the server who responded to the BOOTP request, using TFTP (RFC783) (Trivial File Transfer Protocol) and the pathname returned in the BOOTP response (see the *UNIX Programmers Manual* entries for **tftp** and **tftpd**).

At this point, the PROM boot code starts the kernel running.

Startup—The Mach Kernel

As you may remember from the discussion about local booting, the Mach startup procedure involves:

- initializing hardware (since the kernel makes no assumptions about the state of the machine when it is started, it resets every hardware device)
- initializing kernel data structure
- starting kernel threads
- mounting the root file system

The first three steps are the same as in booting from a local disk. Since NFS is used for all network file system activity (including diskless operation), when the root device is a network interface (i.e. the Ethernet) we need to do the following in order to complete the final step, mounting the NFS root filesystem:

- Configure the Ethernet interface. The kernel uses BOOTP (just like the ROM monitor) to discover its IP address from its Ethernet hardware address (see the *UNIX Programmers Manual* entry for **bootpd**).

- Discover the system's hostname. The kernel uses the BOOTPARAMS "WHOAMI" RPC to broadcast the local IP address. Some BOOTPARAMS server will return the local hostname (and YP domain). Using BOOTPARAMS allows caching the identity of the BOOTPARAMS server (see the *UNIX Programmers Manual* entry for **bootparam**, **bootparams** and **bootparamd**).

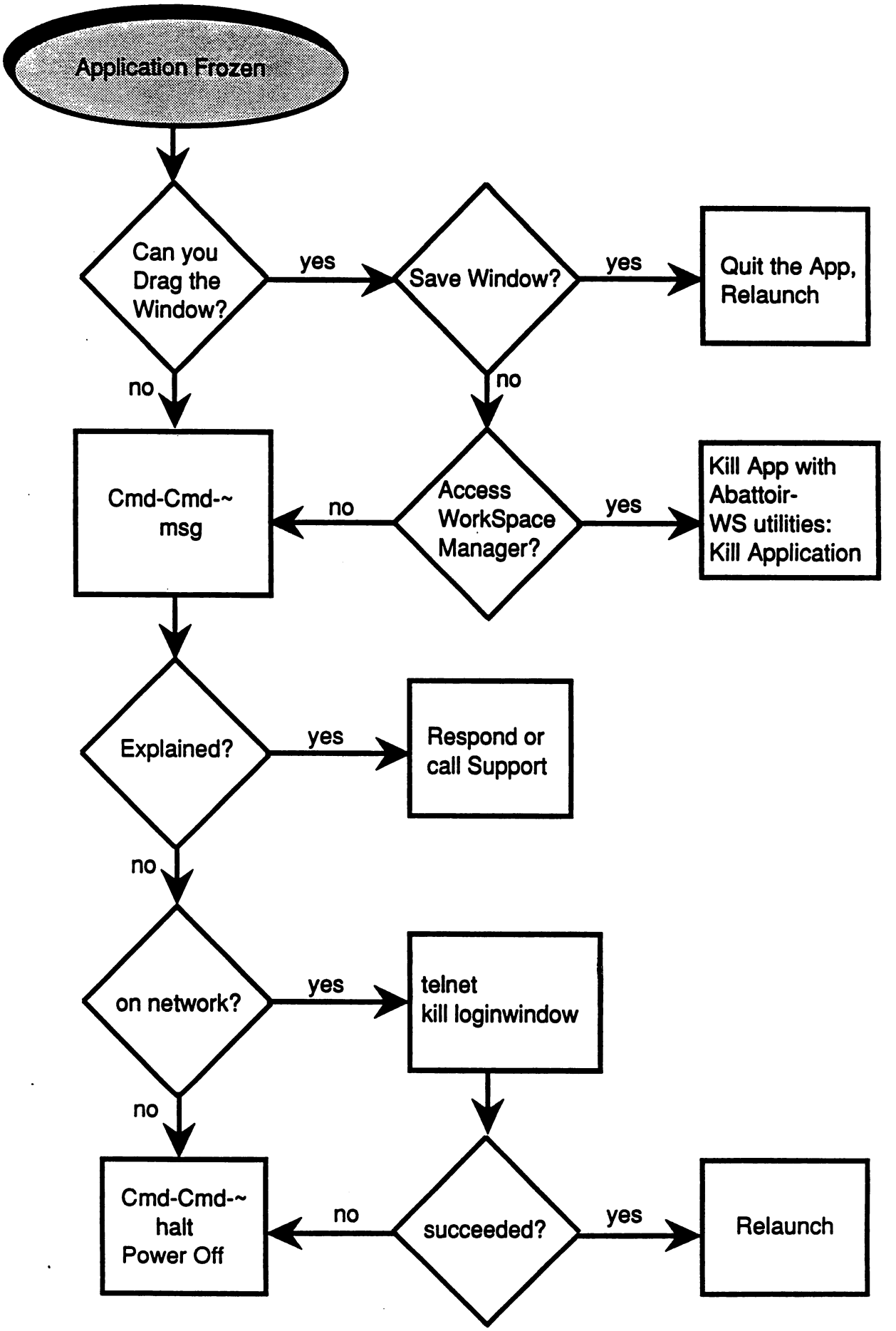
- Find out where the root partition is. The BOOTPARAMS "GETFILE" RPC is used to find out the server name, server IP address and path of our root file system.

- Get an NFS filehandle for the root file system. The MOUNT "MNT" RPC is used to convert a pathname into an NFS filehandle (see the *UNIX Programmers Manual* entry for **mount**).

- Mount this NFS file system on "/". When booting off the network, two file systems are mounted by the kernel. Not only is "/" mounted, the "private" partition is also mounted. This allows the isolation of the files that are private to each system (i.e. **fstab**) and minimizes the amount of file duplication on NeXT fileservers.

Once the file system is set up, the kernel starts the **init** program, which controls bringing the system up in multi-user mode.

On-Screen Trouble-Shooting



Shut Down Procedures

Shutting down the NeXT way: Before shutting down the machine, you should quit each and every application that is running, so that any open files will be saved to disk. Once this is done, simply press the **Power** key. A panel will appear, warning you to save all files by quitting all active applications. You may then select **Power Off.** This is a clean shut down.

Shutting down the UNIX way: If you can become the superuser, log in and launch either **Terminal** or **Shell** and switch user to the super user. From here, you can issue the standard UNIX **shutdown** or **halt** commands. This will leave you in the ROM Monitor with the **NeXT>** prompt. From there you can press the **Power** key, which will give you a prompt asking if you really wish to power off. Respond with a **y** to turn off the power.

If you cannot log in because the Workspace Manger is tossing you back out, you can enter **console** at the loginwindow and get a standard UNIX login prompt. You can now log in and go through the above steps.

These are all clean shut downs.

Returning to the NMI monitor: If you cannot get anywhere because PostScript is locked up, or can't become the superuser, you must resort to keystroke shutdown. Hold both **Command** keys down while typing the keypad tilde key. This will present you with the NMI mini monitor prompt, **nmi>** . From this prompt, type **halt.** After the monitor has synched your disk(s) you will be left in the ROM monitor, from which you can use the **Power** key. This, too, is a clean shut down.

If entering **halt** does not work, you can enter **monitor.** This will take you directly to the ROM monitor, but it is *not* a clean shut down.

If all else fails: If none of the above mechanisms are available to you, you can do a hard reset of the CPU by pressing **Alternate-Command-*** (that's the left **Alternate** and **Command** keys and the numeric keypad ***** key). This will not power off the machine, but it will restart everything. Or, you can pull the plug. (Obviously not a clean shut down.) Be warned—both of these two procedures are potentially damaging to an optical disk if the drive is in the middle of a write operation.

Tracking System Error Messages

On NeXT Computer Systems, as in common 4.3 BSD UNIX systems, error messages are directed to the files specified in the file **/etc/syslog.conf** which, on our system, is a symbolic link to **/private/etc/syslog.conf**

In addition, all kernel error messages are captured by the NMI mini-monitor's message buffer. This summary describes three useful methods for accessing system error messages.

Viewing a History of Error Messages: To view a history of error messages, use the mechanism of your choice to examine the contents of

/private/adm/messages

or use the more familiar name, **/usr/adm/messages** (**/usr/adm** is a symbolic link to **/private/adm**).

This file (and other files located in the **adm** directory) grow and are truncated through the **weekly** or **monthly** script in **/usr/adm** (set up to run automatically by **cron**).

Capturing Error Messages as They Are Generated: To capture system error messages as they are generated, select **Utilities** from the Workspace Manager's main menu, drag into the **Utilities** sub-menu, and select **Console....** This will result in the display of a console window. You can place this window anyplace you like, or miniaturize it if you prefer. This technique is commonly used when you expect error messages to be generated (i.e. when you are attempting to correct ECC errors).

Viewing Error Messages That Were Generated During the Current Boot Session: To view system error messages that were generated during this boot session, enter the **msg** command from the NMI mini-monitor. This is most useful when you can't access the messages file.

NAME

lpc – line printer control program

SYNOPSIS

/usr/etc/lpc [command [argument ...]]

DESCRIPTION

Lpc is used by the system administrator to control the operation of the line printer system. For each line printer configured in the printcap(5) database *lpc* may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer dameons.

Without any arguments, *lpc* will prompt for commands from the standard input. If arguments are supplied, *lpc* interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing *lpc* to read commands from file. Commands may be abreviated; the following is the list of recognized commands.

? [command ...]

help [command ...]
: Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort { all | printer ... }
: Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by *lpr*) for the specified printers.

clean { all | printer ... }
: Remove any temporary files, data files, and control files that cannot be printed (i.e., do not form a complete printer job) from the specified printer queue(s) on the local machine.

disable { all | printer ... }
: Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by *lpr*.

down { all | printer } message ...
: Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message doesn't need to be quoted, the remaining arguments are treated like *echo*(1). This is normally used to take a printer down and let others know why (lpq will indicate the printer is down and print the status message).

enable { all | printer ... }
: Enable spooling on the local queue for the listed printers. This will allow *lpr* to put new jobs in the spool queue.

exit

quit
: Exit from lpc.

restart { all | printer ... }
: Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. *Lpq* will report that there is no daemon present when this condition occurs. If the user is the super-user, try to abort the current daemon first (i.e., kill and restart a stuck daemon).

start { all | printer ... }
: Enable printing and start a spooling daemon for the listed printers.

status { all | printer ... }
Display the status of daemons and queues on the local machine.

stop { all | printer ... }
Stop a spooling daemon after the current job completes and disable printing.

topq printer [jobnum ...] [user ...]
Place the jobs in the order listed at the top of the printer queue.

up { all | printer ... }
Enable everything and start a new printer daemon. Undoes the effects of *down*.

FILES

/etc/printcap	printer description file - if NetInfo is not running
/usr/spool/*	spool directories
/usr/spool/*/lock	lock file for queue control

SEE ALSO

lpd(8), lpr(1), lpq(1), lprm(1), netinfo(5) printcap(5)

DIAGNOSTICS

?Ambiguous command	abreviation matches more than one command
?Invalid command	no match was found
?Privileged command	command can be executed by root only

Shorting NVRAM in Case of Forgotten Hardware Password

The Hardware Password, a new feature introduced with the 1.0 PROM, allows a System Administrator to prevent the unauthorized use of PROM Monitor commands to compromise system security. By default, NeXT Computers have no hardware password but it can be added by typing a capital "P" in the PROM Monitor. The hardware password is stored in non-volatile memory (NVRAM). If the hardware password is forgotten or for some reason needs to be overridden. this can be done by simply wiping out NVRAM. Wiping out NVRAM clears not only the password, but any other ROM Monitor configuration parameters including the system clock.

To return all NVRAM settings to their default values perform the following steps:

1 Remove the CPU board from the cube.

2 Remove the lithium battery from the CPU board. At this point, it is just a matter of time before voltage drains away from the NVRAM wiping it out. It takes quite a long time (over an hour), however, for this step alone to zap NVRAM. The following step assures an instant NVRAM reset.

3 Short pins 10 and 11 on the NVRAM chip by touching the two pins simultaneously with a metal screwdriver or similar tool. (To find the chip, locate the space between the battery socket and the PROM, follow a straight line from this point to the DSP connector on the back of the board, and about 3/4" from the PROM you will find a black, surface-mount chip with 16 leads to the board.) Pins 10 and 11 are located by finding first pin 1. This is the lead which is slightly longer than the rest where it connects to the board. Count down the eight pins on one side of the chip, and up on the other side until you locate pins 10 and 11 on the side opposite pin 1. A touch of these pins together with a metallic tool will short out the NVRAM, clearing its storage. Touching other pins on this chip will not damage it, so if you are not sure which pins are 10 and 11, make sure you make good contact across them all.

4 Insert the battery and place the CPU board back in the cube.

The Configuration Parameters will have defaulted to their "factory" setting which is as follows:

boot device: od
DRAM tests: yes
perform power-on system test: yes
    SCSI tests: no
    loop until keypress: no
    verbose test mode: no
boot extended diagnostics: no
serial port A is alternate console: no
allow any ROM command even if password protected: no
allow boot from any device even if password protected: no
allow optical drive #0 eject even if password protected: no

UNIX Command Summary

Command	Function
mount */dev/filesystem dir*	Attach *filesystem* to the directory *dir*.
umount */dev/filesystem*	Dettach *filesystem* from the directory.
fsck */dev/filesystem*	Check and repair *filesystem*.
disk */dev/diskdev*	Initialize, analyze, repair or eject a disk.
df *filesystem*	Report the amount of disk space used and available in *filesystem* (without arguments, report on all mounted file systems).
du *dir*	Recursively report the amount of disk space used by every directory in and under *dir*.
chown *user file(s)*	Change the ownership of *file(s)* to *user*.
chgrp *group file(s)*	Change the group ownership of *file(s)* to *group*.
su *user*	Begin a new shell process with user id *user*. If *user* is omitted, root is assumed. A password will be required if *user* has one.
ps	Print process statistics.
kill *PID*	Terminate the process identified by *PID*.
lpc	Line printer administration program.
dump *key filesystem*	Create a backup of *filesystem*.
restore *key filesystem*	Restore files from a **dump** archive.

Command	Function
nidump *format domain*	Dump NetInfo data in ASCII to the standard output.
niload *format domain*	Load ASCII data from the standard input into NetInfo format.

Special Files for Serial Connections

The NeXT Computer System provides two physical serial interface ports on the back of the cube. These ports are Mini-Dins following the RS-422 standard and are labeled as serial A and B. These ports can be used with RS-232 with the appropriate pin configurations. Each of these physical ports can be accessed through three different device driver interfaces that provide different types of modem control. The three types of special files associated with these interfaces and the types of modem control they provide are described below.

Note: If you are using communications software that performs error detection and correction, then operating devices at baud rates up to 38400 is perfectly safe. However, if you are communicating without error detection, using baud rates over 4800 runs the risk of lost or garbled data.

Hard-wired Terminals

The special files **/dev/ttya** and **/dev/ttyb** are provided to support directly connected serial terminals. When either of these devices is opened, DTR is asserted. A given port can be opened through either a direct connect interface (**/dev/ttya** or **/dev/ttyb**) or one or both of the modem interfaces described below. Thus, an open through this interface is exclusive of any other opens.

Incoming Modems

The special files **/dev/ttyda** and **/dev/ttydb** are provided to support incoming serial modems (**d** stands for **d**ial-in). When either of these devices is opened (by a getty) DTR is asserted. The driver then waits for the modem to assert DCD (indicating that a remote modem has made contact) before letting the open system call return. When DCD is dropped by the local modem, all processes controlled by the local port are sent a SIGHUP (hangup signal). A port that is opened by either the incoming or outgoing modem interfaces can also be opened by the other.

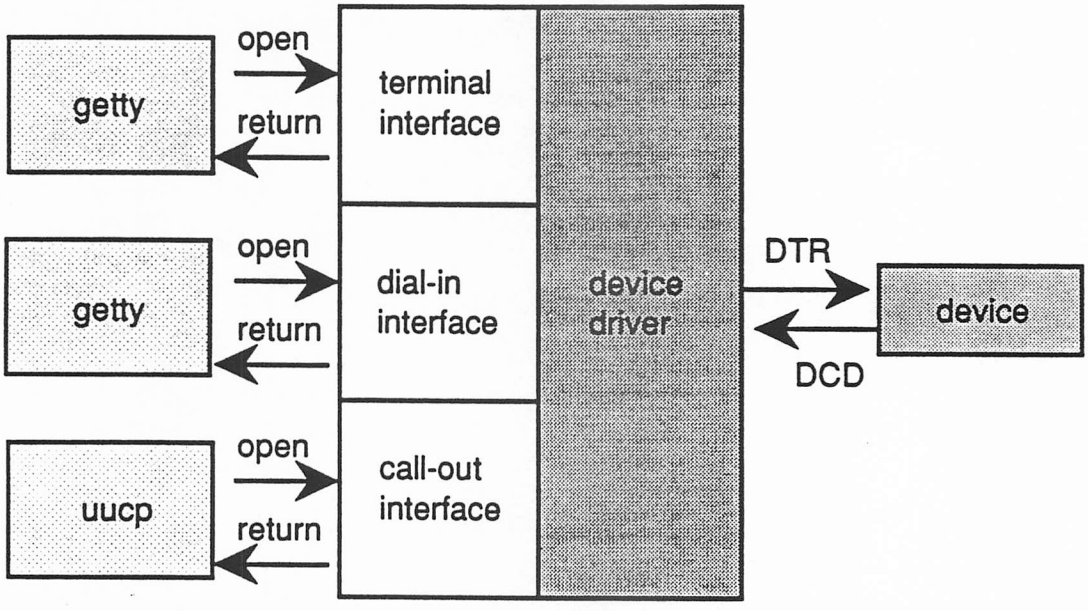
Outgoing Modems

The special files **/dev/cua** and **/dev/cub** are provided to support outgoing auto-dial modems (**cu** stands for call unit) for use with **tip** and **uucp**. When either of these devices is opened the driver asserts DTR and does not wait for the modem to assert DCD. This allows the program that issued the open to give dialing instructions to the modem. When DCD is dropped by the remote modem, all processes controlled by the port are sent a SIGHUP.

Data Files

The file **/etc/ttys** will contain entries for any device files associated with directly connected serial terminals or dial-in modems. Since dial-out modems should not have a **getty** running on their port, entries for these devices will not be found in this data file, but can be found in **/usr/lib/uucp/L-devices**, **/usr/lib/uucp/L.sys** and **/etc/remote**.

Serial Interface Information Flow



Opening Sequence for Terminal Interface

1) getty issues open to /dev/ttyX
2) open returns
3) device driver asserts DTR
4) terminal asserts DCD

Opening Sequence for Dial-in Interface

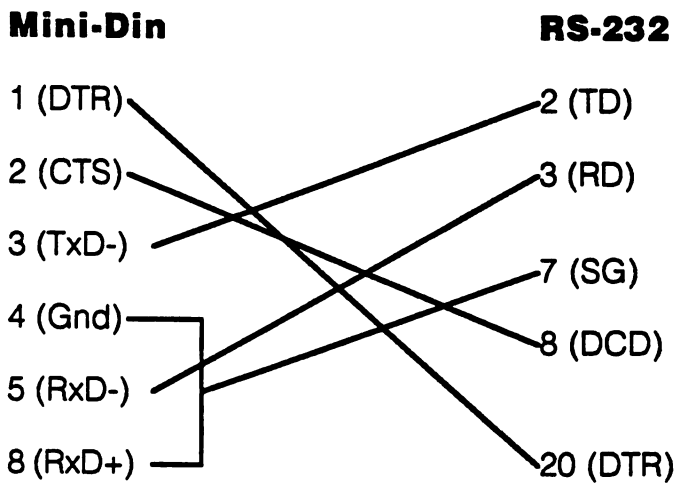
1) getty issues open to /dev/ttydX
2) device driver asserts DTR
3) device driver waits for modem to assert DCD
4) modem asserts DCD
5) open returns

Opening Sequence for Call-out Interface

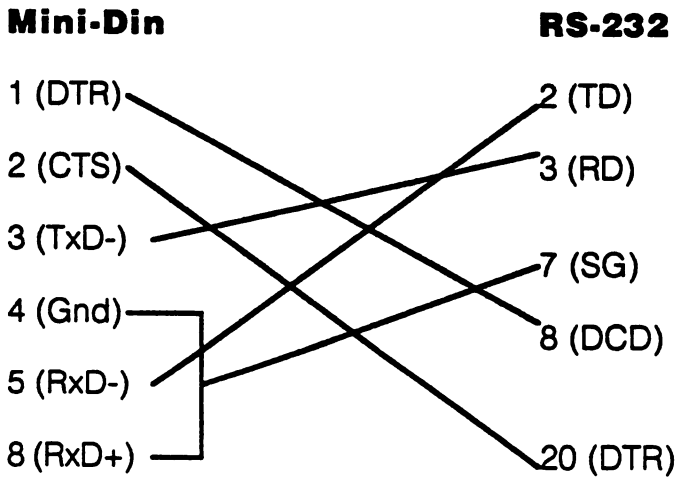
1) uucp issues open to /dev/cuX
2) device driver asserts DTR
3) open returns
4) modem goes through dial and connect sequence
5) modem asserts DCD

Cabling Diagram

Cable for Connecting NeXT Computer to a Modem



Cable for Connecting NeXT Computer to a Terminal
(Null-Modem Cable)



Using a Modem with a NeXT Computer

1 **Connecting a modem:** Use a Mac to Modem cable to connect the modem to the computer's A serial port.

2 **Establishing a connection with serial port A:** Launch **Terminal** and enter

```
tip a9600
```

This command establishes a connection between **tip** and serial port A.

Note: If you get a message saying that the port is already in use, enter the command

```
rm /usr/spool/uucp/LCK..ttya
```

(there is a bug in **tip** that leaves this lock file lying around). Now you can reenter the **tip** command line

There is another bug that leaves the tty port busy; if you get a "tty busy" message, you should kill the tip process.

3 **Getting the modem's attention:** Get the modem to listen to you by entering the appropriate command (for example, FastComms require that you enter **ATQ0**).

4 **Dialing the number:** Dial the number you need by entering the command required by your modem (for Hayes-compatibles, it might be **ATD***<the number>*).

(See the modem manual for any dialing quirks, such as how to put pauses in the dialing sequence, if needed.)

Assuming that all goes well (you entered the correct phone number, the remote modem is turned on, etc.), you will make contact with the target machine which will greet you with a login banner.

5 **Logging in:** Enter a valid user name and password to log in to the remote machine. At this point, you can use the machine as if it were a local machine with a terminal window running.

6 **Logging out:** Once you have completed your session, you can log out from the remote machine by entering the command

```
logout
```

7 **Disconnecting from tip:** After you logout, exit **tip** by entering:

```
~^D
```

(Press the tilde key, then hold down the **Control** key (*not* **Command**), and press **D**.)

NAME

tip, cu – connect to a remote system

SYNOPSIS

tip [**–v**] [*–speed*] system-name
tip [**–v**] [*–speed*] phone-number
cu phone-number [**–t**] [**–s** *speed*] [**-a** *acu*] [**–l** *line*] [**–#**]

DESCRIPTION

Tip and *cu* establish a full-duplex connection to another machine, giving the appearance of being logged in directly on the remote cpu. It goes without saying that you must have a login on the machine (or equivalent) to which you wish to connect. The preferred interface is *tip*. The *cu* interface is included for those people attached to the "call UNIX" command of version 7. This manual page describes only *tip*.

Typed characters are normally transmitted directly to the remote machine (which does the echoing as well). A tilde ('~') appearing as the first character of a line is an escape signal; the following are recognized:

~^D ~. Drop the connection and exit (you may still be logged in on the remote machine).

~c [*name*] Change directory to name (no argument implies change to your home directory).

~! Escape to a shell (exiting the shell will return you to tip).

~> Copy file from local to remote. *Tip* prompts for the name of a local file to transmit.

~< Copy file from remote to local. *Tip* prompts first for the name of the file to be sent, then for a command to be executed on the remote machine.

~p *from* [*to*]
Send a file to a remote UNIX host. The put command causes the remote UNIX system to run the command string "cat > 'to'", while *tip* sends it the "from" file. If the "to" file isn't specified the "from" file name is used. This command is actually a UNIX specific version of the "~>" command.

~t *from* [*to*]
Take a file from a remote UNIX host. As in the put command the "to" file defaults to the "from" file name if it isn't specified. The remote host executes the command string "cat 'from';echo ^A" to send the file to *tip*.

~| Pipe the output from a remote command to a local UNIX process. The command string sent to the local UNIX system is processed by the shell.

~$ Pipe the output from a local UNIX process to the remote host. The command string sent to the local UNIX system is processed by the shell.

~# Send a BREAK to the remote system. For systems which don't support the necessary *ioctl* call the break is simulated by a sequence of line speed changes and DEL characters.

~s Set a variable (see the discussion below).

~^Z Stop *tip* (only available with job control).

~^Y Stop only the "local side" of *tip* (only available with job control); the "remote side" of *tip*, the side that displays output from the remote host, is left running.

~? Get a summary of the tilde escapes

Tip uses the file /etc/remote to find how to reach a particular system and to find out how it should operate while talking to the system; refer to *remote*(5) for a full description. Each system has a default baud rate with which to establish a connection. If this value is not suitable, the baud rate to be used may be specified on the command line, e.g. "tip -300 mds".

When *tip* establishes a connection it sends out a connection message to the remote system; the default value, if any, is defined in /etc/remote.

When *tip* prompts for an argument (e.g. during setup of a file transfer) the line typed may be edited with the standard erase and kill characters. A null line in response to a prompt, or an interrupt, will abort the dialogue and return you to the remote machine.

Tip guards against multiple users connecting to a remote system by opening modems and terminal lines with exclusive access, and by honoring the locking protocol used by *uucp*(1C).

During file transfers *tip* provides a running count of the number of lines transferred. When using the ~> and ~< commands, the "eofread" and "eofwrite" variables are used to recognize end-of-file when reading, and specify end-of-file when writing (see below). File transfers normally depend on tandem mode for flow control. If the remote system does not support tandem mode, "echocheck" may be set to indicate *tip* should synchronize with the remote system on the echo of each transmitted character.

When *tip* must dial a phone number to connect to a system it will print various messages indicating its actions.

VARIABLES

Tip maintains a set of *variables* which control its operation. Some of these variable are read-only to normal users (root is allowed to change anything of interest). Variables may be displayed and set through the "s" escape. The syntax for variables is patterned after *vi*(1) and *Mail*(1). Supplying "all" as an argument to the set command displays all variables readable by the user. Alternatively, the user may request display of a particular variable by attaching a '?' to the end. For example "escape?" displays the current escape character.

Variables are numeric, string, character, or boolean values. Boolean variables are set merely by specifying their name; they may be reset by prepending a '!' to the name. Other variable types are set by concatenating an '=' and the value. The entire assignment must not have any blanks in it. A single set command may be used to interrogate as well as set a number of variables. Variables may be initialized at run time by placing set commands (without the "~s" prefix in a file *.tiprc* in one's home directory). The –v option causes *tip* to display the sets as they are made. Certain common variables have abbreviations. The following is a list of common variables, their abbreviations, and their default values.

beautify
(bool) Discard unprintable characters when a session is being scripted; abbreviated *be*.

baudrate
(num) The baud rate at which the connection was established; abbreviated *ba*.

dialtimeout
(num) When dialing a phone number, the time (in seconds) to wait for a connection to be established; abbreviated *dial*.

echocheck
(bool) Synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted; default is *off*.

eofread
(str) The set of characters which signify and end-of-tranmission during a ~< file transfer command; abbreviated *eofr*.

eofwrite
(str) The string sent to indicate end-of-transmission during a ~> file transfer command; abbreviated *eofw*.

eol
(str) The set of characters which indicate an end-of-line. *Tip* will recognize escape characters only after an end-of-line.

escape

(char) The command prefix (escape) character; abbreviated *es*; default value is "~".

exceptions

(str) The set of characters which should not be discarded due to the beautification switch; abbreviated *ex*; default value is "\t\n\f\b".

force

(char) The character used to force literal data transmission; abbreviated *fo*; default value is "^P".

framesize

(num) The amount of data (in bytes) to buffer between file system writes when receiving files; abbreviated *fr*.

host

(str) The name of the host to which you are connected; abbreviated *ho*.

prompt

(char) The character which indicates and end-of-line on the remote host; abbreviated *pr*; default value is '\n'. This value is used to synchronize during data transfers. The count of lines transferred during a file transfer command is based on recipt of this character.

raise

(bool) Upper case mapping mode; abbreviated *ra*; default value is *off*. When this mode is enabled, all lower case letters will be mapped to upper case by *tip* for transmission to the remote machine.

raisechar

(char) The input character used to toggle upper case mapping mode; abbreviated *rc*; default value is "^A".

record

(str) The name of the file in which a session script is recorded; abbreviated *rec*; default value is "tip.record".

script

(bool) Session scripting mode; abbreviated *sc*; default is *off*. When *script* is *true*, *tip* will record everything transmitted by the remote machine in the script record file specified in *record*. If the *beautify* switch is on, only printable ASCII characters will be included in the script file (those characters betwee 040 and 0177). The variable *exceptions* is used to indicate characters which are an exception to the normal beautification rules.

tabexpand

(bool) Expand tabs to spaces during file transfers; abbreviated *tab*; default value is *false*. Each tab is expanded to 8 spaces.

verbose

(bool) Verbose mode; abbreviated *verb*; default is *true*. When verbose mode is enabled, *tip* prints messages while dialing, shows the current number of lines transferred during a file transfer operations, and more.

SHELL

(str) The name of the shell to use for the ~! command; default value is "/bin/sh", or taken from the environment.

HOME

(str) The home directory to use for the ~c command; default value is taken from the environment.

FILES

/etc/remote	global system descriptions
/etc/phones	global phone number data base
${REMOTE}	private system descriptions
${PHONES}	private phone numbers
~/.tiprc	initialization file.
/usr/spool/uucp/LCK..*	lock file to avoid conflicts with *uucp*

DIAGNOSTICS

Diagnostics are, hopefully, self explanatory.

SEE ALSO

remote(5), phones(5)

BUGS

The full set of variables is undocumented and should, probably, be paired down.

Support Programs

I) Training

A) Systems Support Class

II) Technical Support

A) Administrative

- Two contacts and one alternate

B) email

- ask_next@next.com
- bug_next@next.com
- ...{adobe,pixar}!next!ask_next
- ...{adobe,pixar}!next!bug_next

C) Fax

- (415) 780–3714
- Clearly identify Technical Services and Return Fax Number

D) Telephone

- Available for time critical situations
- 1–800–848NEXT

III) Documentation

A) Technical - shipped post class

B) User - with systems

IV) QA Database

A) Dial-in

B) Weekly printout

V) NeXT Information Exchange

A) Quarterly

B) Documentation or applications

Using Bundled Applications

Introduction

The NeXT Computer comes with a wide variety of bundled applications that make the system productive directly out of the box. This document is provided to give you a very brief introduction to each of the basic applications supplied.

Using Edit

Launch the **Edit** application. Note that it doesn't open a new window for you to work in.

From the **Windows** menu, click **New.** This will cause a new empty edit window to appear on the screen. Notice that it is called **Empty Window** and it's title bar indicates the directory path you would use to find the document. Enter a few line of text by simply typing on the keyboard.

- **Selecting a single character:** To select a single character, place the I-beam cursor in front of (or behind) the letter you want to select. Click and drag through the letter. It will indicate it's selection by a transparent gray field that will enclose the character. This is called *highlighting*.

- **Selecting an entire word:** Selecting an entire word can be accomplished with one of three methods:

  - Use the same technique as when selecting a single character—click at the beginning (or end) of the word and drag to the opposite end. The entire word will highlight.

  - Simply click at the beginning or end of the word (and not drag) and then—while holding down the shift key—click again at the end of the word. The entire word will highlight. This is called *shift clicking* and allows you to extend a selection.

  - Use a short cut—double click anywhere on the word you want to select. The entire word will highlight to show it has been selected.

- **Selecting an entire line:** Selecting an entire line can be done in one of two ways:

  - Use the click and drag technique (as was used on a single character).

  - Click at the beginning of the line you want to select and, while holding down the shift key, click at the end of the line. This will select the entire line and cause it to highlight.

  - Triple click on the line you want to select. The entire line from it's beginning to the carriage return that ends it will be selected and will highlight.

- **Selecting the entire contents of a window:** Selecting the entire contents of a window can be accomplished by using the shift-click technique or the Edit menu's **Select All** command.

- **Changing font attributes:** To change the font attributes of a selection choose the **Font Panel...** choice of the **Font** menu. A panel will appear. This panel will list all the fonts and their associated styles that are available on your particular machine. Choose a **Family**, **Typeface** and **Size** combination. To preview what your choice will look like, click **Preview**. The font panel's text field will change to reflect the choices you made, leaving your document's text unchanged. To apply those choices to the text in your document window click **Set** or hit Return. Notice that changing the selection of a single character changes the entire window of text.

  These text handling techniques demonstrated in Edit are typical of most text based application on the NeXT Computer System.

- **Messaging other applications:** Select a word in your text and choose the **Request** menu. To look up the word in **Webster**, click **Define in Webster** (or use the *command key equivalent* **Command =**). **Webster** will launch, and the word will be defined. To access the **Digital Librarian** or **Quotations**, select the appropriate command from the **Request** menu.

- **Saving a file:** Finally, to save the contents of your window into a file, click in the **Edit** window, then choose **Save** from **Edit**'s **Window** menu. You will be greeted with a *Save panel*, a mini-browser that shows the files in the file system. It will open in you home directory. Type a file name in the Save panel's text field and hit return. If the filename already exists you will be prompted by the system for how to proceed.

Using WriteNow

Launch **WriteNow**. When **WriteNow** opens you will be greeted by a new empty window. It will be called **Untitled-1** and the title bar will indicate that it is currently stored in **/tmp**.

- **Using text handling commands:** All of the text selection techniques you learned in the **Edit** application will work in **WriteNow**, with one minor exception: triple clicking will not work to select an entire line. Instead, you must use click/shift click or click and drag.

  Use the **Font** menu and it's **Font Panel...** choice to change the fonts in your WriteNow document. Note that WriteNow allows a different font attribute to be set for each individual character in the window.

- **Using the ruler line:** To control formatting options like centering, justification and margins, bring **WriteNow**'s ruler on the screen. To do that, choose **Show Ruler** from the **View** menu or use the command key equivalent **Command-r**. The document will redraw itself with a ruler at its top. The ruler commands only work on the paragraph that currently contains the insertion point. To make changes to more than one paragraph at a time, select all the paragraphs that you want to change and then use the ruler to make the changes for all those paragraphs.

  Use the ruler to set margins. Simply drag the downward pointing triangles to reset the left and right margins. A line will be drawn from the margin marker down the length of the document to indicate the position of the new margin. When you release the marker, the text in the affected paragraphs will be redrawn.

  The buttons on the left side of the ruler control text justification. The leftmost button stands for *ragged right*, the next button for *fully justified* text, the next button for *centered* and the rightmost button stands for *ragged left* text.

The other triangles on the ruler are tabs. They can be moved, dragged off the ruler, or replaced. These are, from left to right, *left tab*, *centered tab*, *right tab* and *decimal tab.*

- **Messaging_other applications:** **WriteNow** allows you to make the same requests of **Webster, Librarian** and the **Quotations** applications as did **Edit.** Simply make a selection and choose the **Request** command you would prefer. Remember that **Define in Webster** has a command key equivalent of **Command-=**.

- **Saving to a file:** To save your **WriteNow** document, choose **Save** from the **Window** menu. The **Save** panel will allow you to save your **WriteNow** file in the directory you prefer. You can also choose to save the document in a different format. **WriteNow** supports its own private document format which is the default, *ASCII text* and *RTF text.* The latter two would typically be used to transfer documents to applications on other types of systems. ASCII text loses its formatting, while RTF text maintains the majority of the formatting information you added in **WriteNow.**

Cutting and Pasting Between Applications

Sometimes you need to move information, like text or graphics, from one application to another. This is surprisingly simple.

- **Selecting text to be copied:** First select the information you want to move to another application. If it is text information use the selection techniques described above. To select a graphic you typically single click it. For this exercise, open the file you created earlier in **Edit**, and select a line of text.

  Next, choose **Copy** from the application's **Edit** menu. (You could also use **Cut**, but the selection will be removed from your document.) Either method will cause the indicated text to be placed on the *pasteboard*, a temporary location for all cut and copied objects on the system.

- **Inserting the copied text:** Now, open the file you created earlier with WriteNow.

  Place the insertion point in the location at which you want the new information to be placed. Choose **Paste** from this application's **Edit** menu (or use **Command-v**). The text

that was on the pasteboard now replaces the insertion point. Note that if you made a selection by dragging (or other means) the entire text in that selection will be replaced by the text you paste from the pasteboard.

Using Multimedia Mail

The **Mail** application provides a simple way to communicate with other users on a network, allowing text, voice, graphics and file transfers. Launch this application so that you can experiment with it.

- **Opening a send window:** To send a message to another person on the network, you need to know their *login name*. For this exercise you will send mail to yourself, the user named *me*.

  In the main Mail window, click the **Send** button. A new **Send Window** opens for you to address and fill. In the field that's labelled **To:** type *me*. Fill in the **Subject:** as you wish. The **cc:** field is used to send copies of this message to other logins. Finally, fill in the message field with whatever you feel is an appropriate message.

- **Attaching a voice message:** To add *voice mail* to this message, click **Lip Service**. A panel similar to the controls for a tape recorder comes on the screen. With a microphone attached to the microphone jack on the back of the monitor, click **Record**. Your mike input will be recorded until you click **Stop**. Play back your recording, erase, and re-record until you are mostly satisfied with your message. Then, click the **Edit** button in the **Lip Service** panel. The current sound will be displayed as a *waveform* that you can select using click and drag, and edit using cut, copy and paste (all of the techniques you used earlier for text). A simple application of this ability would be to remove unwanted silence at the beginning and end of you message. Explore the possibilites of editing voice mail until you are happy with you message. Then click **Insert** to include the voice mail in the message you are creating. The voice mail message will be included in your current message and signified by a *lip service icon* in the message text area. Close the **Lip Service** window.

  Notice that the **Deliver** button has changed to include a warning triangle. This symbol indicates that the message you are creating includes features that can only be received by a NeXT computer.

- **Attaching a document:** Other attachments are possible. To include the **Edit** document you made earlier, locate the file in question in the Workspace Manager's Browser, and drag the resulting icon from the Browser's icon well into the **Send Window** of the message you are creating. An icon will appear in the window, indicating the attached file. This same technique applies to other documents as well as entire directories.

- **Delivering mail:** When you are done, click the **Deliver** button. This closes the **Send** window and schedules the mail message for delivery. The message should appear momentarily in your main mail window.

- **Receiving mail:** To read an incoming mail message, simply click the listing in the mail panel. Notice that the listing includes a triangle, indicating that the message has attachments.

  To listen to the voice mail message, double click the lips in the message area to open the tape-recorder-like controls of the **Lip Service** panel once again. Click **Play** to listen to the message. To examine an attached file, double click the associated icon. This will call up the appropriate application to display the file/

- **Messaging other applications:** Note that **Mail** also has a **Request** menu, meaning that you can look up a word in your mail document simply by selecting it and typing **Command-=**.

Using Digital Webster

- **Defining a word:** Once Webster is open, you can simply enter a word in Webster's text field and press **Return** to have it defined for you.

  Once you've defined the word, you may need to better understand some of the words used in the definition. To define a word found in Webster's definition field, place the cursor over the word, hold down the **Command** key and double click. This will bring up the definition of the selected word.

- **Examining pictures:** This is a complete dictionary and as such it includes pictures. Look up *cornice* or *bill* or *donjon*, for example, to see a definition that includes a picture.

To print the picture, choose **Print...** from **Webster**'s menu. Both the definition and the picture will print.

- **Expanding the search:** Sometimes you want to look not for a word's definition, but rather for all definitions that *contain* a certain word. To search in this way, chose **Preferences....** A panel will appear that allows you to change how Webster searches. Click the **Bold Word** button (which searches for the definition of that word) so that it switches to **Other Word** (which will display all definitions that contain that word). Close the **Preferences** window, type *mammal* and hit **Return**. You will be greeted by a series of definitions and pictures that represent all the definitions in **Webster** that contain the word mammal.

Using Digital Librarian

The **Digital Librarian** is a multi-purpose search engine. It supports a range of targets and can be used to search any file on the system. These could include the on-line edition of Shakespeare, your personal files or documentation on the NeXT Computer System. For the following examples, suppose you are writing a program and you need to research how to set a text object as part of the program.

- **Selecting a search target:** Launch the **Digital Librarian**. Click the **NeXT Manuals** icon to make the on-line edition of the documentation your current target. Enter the words "set text" and press **Return** or click on the **Search** button.

- **Examining a reference:** The number of files found which contain the words "set" and "text" are indicated by the number appearing below the text panel. In the lower window these file names are listed for you. Using our example, more than 50 files have been located. Click one of these, and information relative to the file will appear in the bottom window. In this case, it will launch the **WriteNow** Application and display the file, highlighting the first instance of "set." Other instances can be located with the **Find** button from the **Digital Librarian** or with the **Find** option in the **WriteNow** menu.

- **Targeting the search:** Try applying a narrower search. Searching for the term "settext." Notice that only 2 files are found which contain this term.

- **Adding and indexing new targets:** You can add new targets to the **Digital Librarian**. These targets can be either individual files or entire directories. Drag the icon for either of the two documents you created earlier from the **Workspace Manager**'s Browser to the area in the **Digital Librarian** where the other icons are. This is the *target area*.

  Directories are indexed. Other documents are searched *in toto*. As soon as you release the icon in the target area you will either be prompted by the **Digital Librarian** to create an index for the new target (a directory), or informed that the file cannot be indexed. (This process is only necessary once.) Agree to this process and the files in your directory will be indexed so that the Librarian can search them very quickly. After a few moments (depending on the size of the target files) you will be able to search this target. Look up something that should be there and see what happens when you search for something that should not be there.

- **Removing targets:** The menu for the Digital Librarian has a Target selection which allows you to remove targets, as well as update an index.

- **Searching multiple targets:** By the way, you can shift select multiple targets to be able to search for the same thing in a number of different places at the same time. For example, you may want to search for a command in both the NeXT reference manuals and the release notes.

Conclusion

At this point, you probably don't consider yourself an expert in any of the applications presented here. However, you have hopefully gained enough insight to give you the courage and curiosity to explore further. There's nothing like practicing with an application for developing expertise. Don't forget, it's also perfectly okay to enjoy yourself along the way.

Using Mathematica

Mathematica is a general system for doing mathematics on computers. It attempts to provide for mathematicians the sort of simplification, organization and amplification that word processors provide for writers.

Getting Familiar with Mathematica

You can do most operations in Mathematica based on your intuition of how it "should" work. Mathematica closely follows standard mathematical conventions. It also reflects traditional programming languages, especially C, and so a familiarity with programming will also lead to trustworthy intuitions about how things work. Mathematica reflects the standard user interface of the NeXT computer so you can apply your NeXT intuitions just as effectively.

- Launch the Mathematica application, using your favorite method.

Once the program is running, you will be greeted with an empty window and a menu panel. The window is similar to a standard application window. You will see the close box in the upper right hand corner and the miniaturize box in the upper left. Along both the left side and the bottom of the window are scroll bars.

- Move your cursor up to the menu panel and click **Help**. This will reveal a sub-menu. Note the **Why The Beep?** command and the **Help Cursor** command. You'll find these two commands very helpful while learning this program.

The help cursor is a way to get context sensitive help while using the program. To use this feature, click the menu selection **Help Cursor**. You'll see the cursor change to a question mark. Now click any menu choice you would like to know more about. Mathematica will bring up its help file selection for the item you choose.

As with many menu commands, **Help Cursor** has a keyboard alternative: **Command /**. You can also get to the help panel through the menu selection **Info...**.

- Use the help commands to explore each of the menus in Mathematica.

Working with Mathematica

- Click somewhere within the borders of the main Mathematica window and enter:

```
- 23
```

- Now press the **Enter** key on the numeric keypad.

In order to get Mathematica to evaluate an expression you must press the **Enter** key on the numeric keypad. Note that this is different than the **Return** key. In this program, the **Return** key only produces the same effect as the **Enter** key if you also hold down the **Shift** key when you hit **Return**.

After you pressed **Enter** the first time during a Mathematica session, you will experience a brief delay while Mathematica loads the local kernel. Mathematica consists of two parts: a kernel that does the actual calculation and a front-end that supports the local user interface. For people familiar with UNIX, this is similar to the relationship of a shell program to the UNIX kernel. Until you pressed **Enter**, you were only interacting with the interface.

If you find yourself sitting there, wondering why Mathematica is taking so long, check the window's title bar to see if it says **Running**. If not, you might want to check to see if you have inadvertently pressed **Return** to conclude your input, (which simply gives you a carriage return to allow formatting of the input lines on the screen) rather than enter, which asks Mathematica to evaluate your input.

Mathematica will type back:

```
In[1]:=
                23

Out[1]:=
                23
```

The first **23** is simply your input to the program (as indicated by **In[1]**), echoed in a form that makes it easy to identify and to refer to later. The second **23** is Mathematica's best guess at how to evaluate the expression. Notice no beep, no error condition. While simplistic, this exchange is totally legal. Notice also there's now a bracket on the righthand margin. This bracket encloses the two expressions, each of which has its own sub-bracket. This is how Mathematica indicates *cells*, which are the basic unit of organization in a Mathematica session.

- Now type:

```
23 / 0
```

Don't foget to end with the `Enter` key. The slash means divide. After a moment the program will respond with a message and an evaluation of your expression. Dividing by zero, while not mathematically legal, simply causes Mathematica to print a note about "complex infinity." Still no beep.

- Now enter:

```
23 [
```

Note that we have an open bracket without a close bracket. Brackets are how Mathematica encloses variables to be evaluated. How do you think Mathematica will evaluate this syntax error?

Now we hear a beep. If its clear to you what is wrong you can simply go ahead and fix it. Mathematica will not disturb your work with more interference than that. The cursor will be left where Mathematica got stumped by your expression.

- To get more information, simply click on **Why the Beep?** and you will get a panel explaining the problem from Mathematica's point of view.

Now that you have a feel for the program, try to do something (slightly) less trivial.

- Click in the Mathematica window below the existing cells and enter the following:

```
2 + 2
```

Mathematica returns:

In[3]:=

```
2 + 2
```

Out[3]:=

```
4
```

- Now lets take that result and square it.

```
% ^ 2
```

To Mathematica the percent sign has a special meaning. It is used to refer to the result of the previous evaluation.

The response:

In[4]:=

```
% ^ 2
```

Out[4]:=

```
16
```

- Now lets assign that intermediate result a name to make it easier to work with.

```
square4 = %
```

And Mathematica duly responds with:

In[5]:=

```
square4 = %
```

Out[5]:=

```
16
```

Now, unless you explicitly rename or unassign this variable, the name *square4* will mean 16 until the end of your Mathematica session. Note that variable names are always lower case to distinguish them from built in Mathematica objects which are identified with names beginning with capital letters.

- Lets take our result and manipulate it a little. Type:

```
Sqrt[square4]
```

And you'll see:

In[6]:=

```
Sqrt[square4]
```

Out[6]:=

```
4
```

Not surprisingly, this Mathematica object performs square roots. Note its name begins with a capital letter. Would we have gotten the same result if we had referred to the previous result in a different way? Use % to refer to the last result, %% to refer to the next to last result, or *%n* where *n* refers to the line number of the result. You should find that regardless of how you refer to the result, whether explicitly, as a named variable, or as a line number reference that the underlying evaluation produces the same result.

To wrap up this brief sampling of Mathematica, let's plot a graphic on the screen.

- Click in the Mathematica window and type the following:

```
Plot3D[ Sin[x] Sin[3y], {x, -2, 2}, {y, -2, 2} ]
```

Note that all the names of Mathematica objects start with a capital letter and that all the brackets are matched. The result is a PostScript display of the plot of the entered formula.

To finish this Mathematica session, you might want to save what you've done so far into a notebook. Click **Window** in the main menu panel. This will reveal the **Window** sub-menu. Click **Save** from this menu and name your file **First Session**.

For further exploration, on your disk in the directory **NeXTLibrary** you will find a **MathematicaNote** directory that contains a file called **A Tour of Mathematica**. This is another Mathematica notebook, or a set of Mathematica expressions, evaluations and explanatory text. Each topic is organized in a series of hierarchical cells. To open a collapsed cell, click on the cell bracket in the right margin of the topic you want to explore. Expandable cells are designated by a special bracket which looks like its lower end has been folded up. These cells can be opened by a double-click on the bracket itself. When you are done, double-clicking again will collapse that section.

You might also wish to explore the Demonstration application called **RealTimeAlgebra** that can be found in the directory **/NextDeveloper/Demos**.

Introduction to Interface Builder

Introduction

Traditionally, a programmer's first task in a new environment has been to get some input, like the words "Hello, World", to appear on an output device, like the screen or on paper. This gives one a sense of the friendliness of the environment, the type of tools used, the power of those tools and should ideally provide an enticement to continue using the environment.

In using languages like C and Pascal this exercise typically produces five or ten line programs. On the other hand, the complexity of a graphic user interface environment presents a major barrier to doing anything interesting in five or ten lines of code.

Enter Interface Builder. Interface Builder is a tool for creating user interfaces that also provides support for the programming code that underpins those interfaces. This exercise will give you some friendly exposure to this tool.

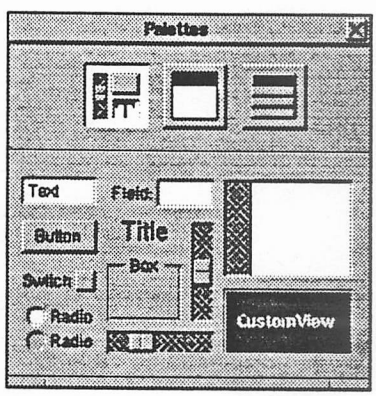
Interface Builder

Launching Interface Builder: To start the Interface Builder application, either double click its icon in the dock (it has a screwdriver above two screw heads) or select the **NextApps** directory in the Workspace Manager's Browser, select **InterfaceBuilder** and double click on the icon in the icon well. The program's icon will highlight while it is launching.

Examining the objects on the screen: Take a look at what you get. When Interface Builder opens you will see two new objects on the screen.

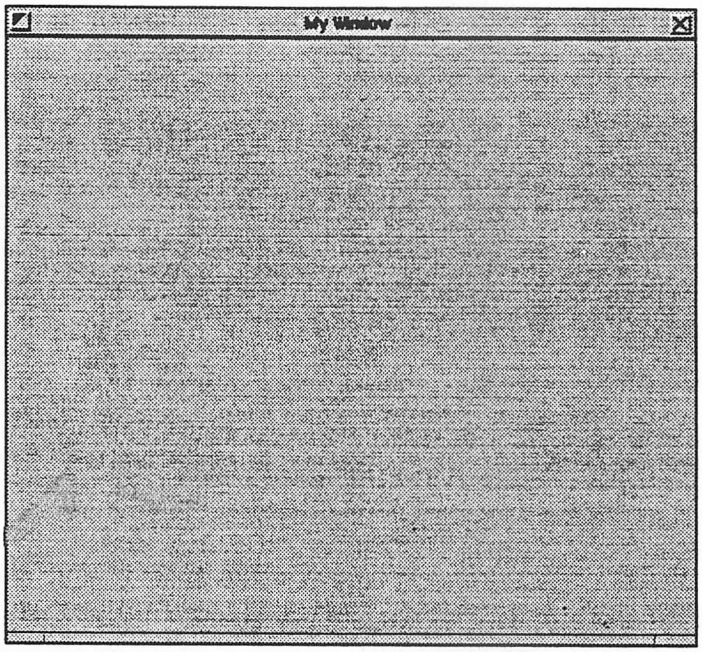
- Interface Builder's Main Menu (labelled **IB**) is in the upper left corner of the screen.

- To the right there is a window called **Palettes.**



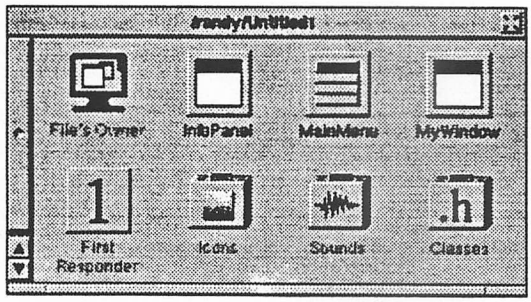
Creating a new application: Under the **File** menu choose **New Application.** Several things happen.

- In the center of the screen is your new window, which is empty.



- Beside it is your menu.

There is also another window which looks like an icon view in the Workspace Manager's Browser. This is the *File window*. This contains icons that represent the files that will make up your new application.



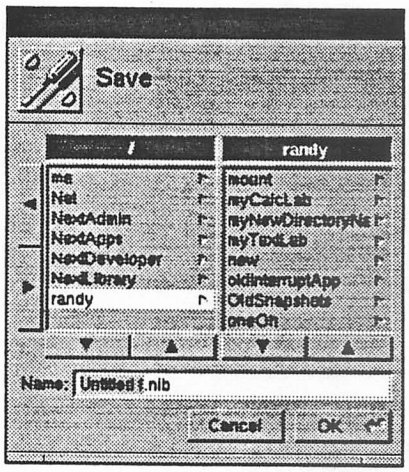
Notice that the close box on this window indicates that the file has not been saved.

See if you can identify which files belong to the various screen objects. Are there some objects in the File window that don't have a corresponding screen object? Are there any screen objects that don't have a corresponding file icon?

What happens when you click the close box of MyWindow? Can you find a way in the File window to bring the screen object back in to view?

Creating a new directory and saving your work: You know by the partial X that your interface file is not yet saved. Click on the **File** menu and choose **Save.**

You will be greeted with a *Save panel*, a mini-browser that shows the files on the disk.



Notice the Save panel comes up open to your home directory with a default filename of **Untitled.nib.** The file extension for Interface Builder files is **.nib** (which stands for NeXT Interface Builder). This file will contain the objects that you specify for your interface (objects like windows and menus).

You need to create a new directory for your interface file. You can do this as you save. In the text field of the Save panel, type a new directory name followed by a name for your interface file:

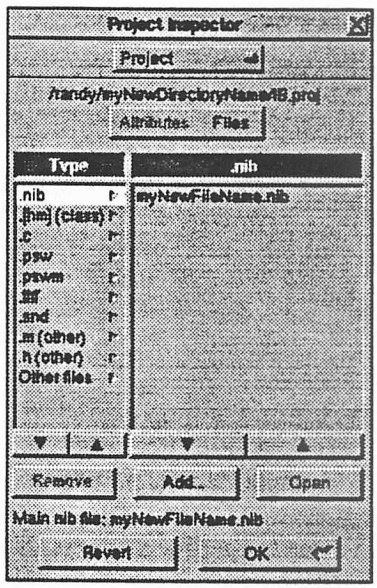
```
Project1/MyFile
```

Click **OK**. You will be informed (by the **Bad Path** panel) that the path does not exist and asked if you want to create it. You do indeed. Click on **Create** to create the new directory and save your file in it.

Note that the close box on the file window changes to reflect the fact that the file has been saved and its title bar reflects the file name and path you've chosen. This indication can be useful when you have multiple interface files open in Interface Builder.

Interface Builder requires that you keep all the files for each of your projects in separate directories.

Creating a project file: Interface Builder also allows you to control the other files that you use in a programming project (in addition to your interface file). Under the **File** menu select **Project...** to create a *project file* for your application. You will first be greeted by the *Inspector panel* telling you that "There is no project file ..." and that you should "Click OK to create one". Go ahead and do as you are told. This will put the panel in the mode for inspecting project files.



Notice that the **Project Inspector** comes up indicating that the project has a file of the type **.nib**—an interface file called **MyFile.nib**.

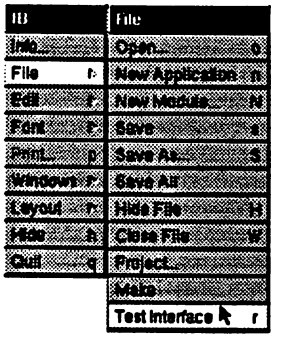
Through this project management tool, Interface Builder keeps track of the various components that you are using to create an application. Note that there is only one project per directory.

Creating a new directory, saving your interface in it and then creating a new project file are three essential steps to creating a new application in Interface Builder.

You need to do this for each application before you create any other files—so that those files can be included in your application, tracked by Interface Builder and kept updated.

To review, there is an interface file, and its included in your project. This should be no surprise—you explicitly created and saved the interface file.

Exploring the difference between Test mode and Build mode: So far you have been in Interface Builder's Build mode. On the **File** menu there is a choice called **Test Interface.** Choose that item.



A lot will happen—all of Interface Builder's windows will hide and your new application's windows will be left on the screen. Interface Builder's icon will change to look like a big switch.

- Try moving and resizing the window.
- Try the **Info...** choice from your menu and examine the Info panel.
- Note that the **Hide** command works, but to *unhide*, double click on the Interface Builder's switch icon (as your program doesn't have an icon yet).

In Test mode you get a simulation of how your application's interface will act when someone uses it.

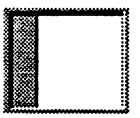
In Build mode you change the way your application's interface looks and what it contains.

To return to Interface Builder's Build mode at any time, double click the switch icon or click **Quit** on your application's menu.

Building the Interface

Locate the **Palettes** window. It's on the upper right side of the screen.

Adding a place for your text: The white field with a gray bar along its left side is an object that displays scrolling text. Drag it into your application's window. Your application will use this to hold and display text.



Now that you have an instance of a scrolling text field in your application's window, resize it by selecting it (clicking it), grabbing a *handle* (one of the little squares that appeared around the object when you selected it) and dragging it to a size you prefer. Note that it stays within the boundaries of the enclosing window.

Testing the scrolling text: Now, try the Test mode once again. Click in the first line of the text field. You should see a blinking insertion point. Try typing some text, like "Hello, World".

- You should be able to select text with the mouse or with the **Select All** option of your Edit menu.

- The selected text can be cut, copied and pasted.

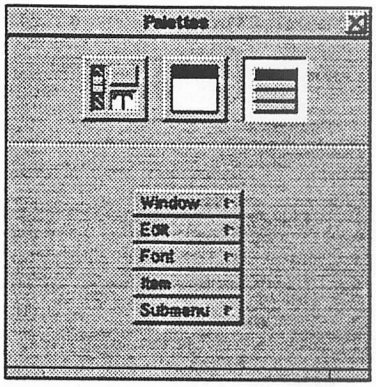
- Note that anything cut or copied from this application can be pasted into any other application that deals with text, and vice versa.

When in Test mode, this is how your application will appear to the person using it. Return to Build mode. Note that any entered text has disappeared.

Save your interface file again by choosing the **Save** command from the **File** menu.

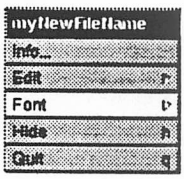
As you can tell from the Test mode, the text object already knows a lot about basic text handling. It can wrap text to a view, it can scroll and it can select text via user input. It also supports text editing via cut, copy and paste. Let's extend the functionality of this little application by adding a menu item to control font attributes.

Adding the Font Menu: Locate the **Palettes** window once again. Click on the far right selector which stands for menus.



This will reveal a menu with some default menu cells. Drag the **Font** menu cell to your application's menu.

Place it just above **Hide** and release.



Check in the File window of your application. Were any new icons added as a result of dragging the Font menu item into the interface?

Testing the Font Menu: Now go into Test mode once again and check the font commands you just added. Note that the changes you make for any character in the window are reflected throughout the window, not just in the selection, (just like the Edit applications).

Leave the Test mode. Once more save your interface file.

Modifying the Interface

Inspecting the Inspector: Let's take a look at the Inspector panel while it's out. This panel will allow you to change and initialize many attributes of many objects. Just below it's title bar is a pop-up list that will allow you to choose which mode you are in. The modes are:

- Attributes—controls colors, styles of objects; their general on screen appearance, ranges and basic state of interface objects.

- Connections—allows you to connect an object to other objects in two ways:

- Autosizing—controls the behavior of objects that are contained in other objects that resize (e.g. a text field inside a window that is being resized by the user).

- Miscellaneous—controls size and placement of certain interface objects as well as their names.

- Class—allows a programmer to describe the outlets and actions of a new class of object, one that might inherit behavior from another class (like a Window object) and add new behavior (like *closeQuick!*).

- Project—allows the management and automatic update of the various files that are the component parts of an application and of its preferred state.

Note that the Inspector's context is the currently selected object. You change that context by selecting a new object—either clicking it's icon in the Files window or clicking the object on the screen.

Personalize the Info panel: Let's touch on the Info panel. Double click its icon in the Files window to bring it on screen so you can edit it.

- By selecting the objects it contains you can modify them directly—or use the Inspector to change their values or appearance.

- To select text, double click the text you want to change.

- Use Interface Builder's **Font** menu to modify your text's attributes.

Give yourself credit as the author of this program in your Info panel by selecting an existing text field (by double clicking) and replacing the resulting selection with your name. Make any other changes you think appropriate. Experiment & be creative.

Use test mode to observe your changes: Try the test mode once again.

- See how your content and style changes are reflected in the test version of your program.

- See if the on screen look is what you want.

- Test the interface for function.

- Try resizing the window.

What happened to the field when the window was resized? If this behavior is not what you'd prefer, can you find a way (using the Inspector) to change how autosizing works?

Exercise the interface, modify it and test it again until you are satisfied.

> *The cycle of exercise, modify and test is one that you will often use in perfecting a user interface.*

Save all your work using the **Save** command in the **Files** menu of Interface Builder.

Compiling the Program

So, you have an interface and some underlying code. Let's compile it.

From the Files menu choose **Make.**

Using the Make utility: This command will cause a command line interface application called **Shell** to run. A command from the **make** utility (a system utility program which controls compiling programs) will be typed for you in the **Shell** window, once the current directory is changed to point to the directory that contains your new interface and project. **Make** will use the interface file you created, the *main()* routine Interface Builder wrote for you automatically and the other information you gave the **Project Inspector** to create an executable version of the program you have been working on.

Testing the compiled version: Once the make process has completed and there are no errors (there shouldn't be) test the compiled version of your application. Launch it from the Workspace Manager's Browser. The newly created application will be called **My File .debug.** It will display a generic application icon.

Once more, give the program a thorough testing.

Congratulations on creating your first program. Pat yourself on the back.

NAME

arp – address resolution display and control

SYNOPSIS

arp *hostname*
arp -a [*mach*] [*kmem*]
arp -d *hostname*
arp -s *hostname ether_addr* [**temp**] [**pub**] [**trail**]
arp -f *filename*
arp -e *filename*

DESCRIPTION

The *arp* program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (*arp*(4p)).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation. With the **-a** flag, the program displays all of the current ARP entries by reading the table from the file *kmem* (default /dev/kmem) based on the kernel file *mach* (default /mach).

With the **-d** flag, a super-user may delete an entry for the host called *hostname*.

The **-s** flag is given to create an ARP entry for the host called *hostname* with the Ethernet address *ether_addr*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word **temp** is given in the command. If the word **pub** is given, the entry will be "published"; i.e., this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host.

The **-f** flag causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

hostname ether_addr [**temp**] [**pub**] [**trail**]

with argument meanings as given above.

The **-e** flag causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

ether_addr hostname [**temp**] [**pub**] [**trail**]

with argument meanings as given above. Note that this flag is the same as the **-f** flag except that the first two fields in the file to be read are reversed. This format is suitable for reading the /etc/ethers file used by the yellow pages.

SEE ALSO

inet(3N), arp(4P), ethers(5), ifconfig(8C)

NAME

netstat – show network status

SYNOPSIS

netstat [**–Aan**] [**–f** *address_family*] [*system*] [*core*]
netstat [**–himnrs**] [**–f** *address_family*] [*system*] [*core*]
netstat [**–n**] [**–I** *interface*] *interval* [*system*] [*core*]

DESCRIPTION

The *netstat* command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented. The first form of the command displays a list of active sockets for each protocol. The second form presents the contents of one of the other network data structures according to the option selected. Using the third form, with an *interval* specified, *netstat* will continuously display the information regarding packet traffic on the configured network interfaces.

The options have the following meaning:

–A With the default display, show the address of any protocol control blocks associated with sockets; used for debugging.

–a With the default display, show the state of all sockets; normally sockets used by server processes are not shown.

–h Show the state of the IMP host table.

–i Show the state of interfaces which have been auto-configured (interfaces statically configured into a system, but not located at boot time are not shown).

–I *interface*
Show information only about this interface; used with an *interval* as described below.

–m Show statistics recorded by the memory management routines (the network manages a private pool of memory buffers).

–n Show network addresses as numbers (normally *netstat* interprets addresses and attempts to display them symbolically). This option may be used with any of the display formats.

–s Show per-protocol statistics.

–r Show the routing tables. When **–s** is also present, show routing statistics instead.

–f *address_family*
Limit statistics or address control block reports to those of the specified *address family*. The following address families are recognized: *inet*, for **AF_INET**, *ns*, for **AF_NS**, and *unix*, for **AF_UNIX**.

The arguments, *system* and *core* allow substitutes for the defaults "/mach" and "/dev/kmem".

The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form "host.port" or "network.port" if a socket's address specifies a network but no specific host address. When known the host and network addresses are displayed symbolically according to the data bases */etc/hosts* and */etc/networks*, respectively. If a symbolic name for an address is unknown, or if the **–n** option is specified, the address is printed numerically, according to the address family. For more information regarding the Internet "dot format," refer to *inet*(3N). Unspecified, or "wildcard", addresses and ports appear as "*".

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit ("mtu") are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route ("U" if "up"), whether the route is to a gateway ("G"), and whether the route was created dynamically by a redirect ("D"). Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface. The refcnt field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

When *netstat* is invoked with an *interval* argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface found during autoconfiguration) and a column summarizing information for all interfaces. The primary interface may be replaced with another interface with the *–I* option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

SEE ALSO

iostat(1), vm_stat(1), hosts(5), networks(5), protocols(5), services(5), trpt(8C)

BUGS

The notion of errors is ill-defined. Collisions mean something else for the IMP.

NAME

ftp – ARPANET file transfer program

SYNOPSIS

ftp [–v] [–d] [–i] [–n] [–g] [host]

DESCRIPTION

Ftp is the user interface to the ARPANET standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

The client host with which *ftp* is to communicate may be specified on the command line. If this is done, *ftp* will immediately attempt to establish a connection to an FTP server on that host; otherwise, *ftp* will enter its command interpreter and await instructions from the user. When *ftp* is awaiting commands from the user the prompt "ftp>" is provided to the user. The following commands are recognized by *ftp*:

! [*command* [*args*]]
Invoke an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any *ntrans* or *nmap* setting. File transfer uses the current settings for *type*, *format*, *mode*, and *structure*.

ascii Set the file transfer *type* to network ASCII. This is the default type.

bell Arrange that a bell be sounded after each file transfer command is completed.

binary Set the file transfer *type* to support binary image transfer.

bye Terminate the FTP session with the remote server and exit *ftp*. An end of file will also terminate the session and exit.

case Toggle remote computer file name case mapping during **mget** commands. When **case** is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.

cd *remote-directory*
Change the working directory on the remote machine to *remote-directory*.

cdup Change the remote machine working directory to the parent of the current remote machine working directory.

close Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

cr Toggle carriage return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete the file *remote-file* on the remote machine.

debug [*debug-value*]
Toggle debugging mode. If an optional *debug-value* is specified it is used to set the debugging level. When debugging is on, *ftp* prints each command sent to the remote machine, preceded by the string "-->".

dir [*remote-directory*] [*local-file*]
Print a listing of the directory contents in the directory, *remote-directory*, and, optionally, placing the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, or *local-file* is -, output comes to the terminal.

disconnect
A synonym for **close**.

form *format*
Set the file transfer *form* to *format*. The default format is "file".

get *remote-file* [*local-file*]
Retrieve the *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current *case*, *ntrans*, and *nmap* settings. The current settings for *type*, *form*, *mode*, and *structure* are used while transferring the file.

glob Toggle filename expansion for **mdelete**, **mget** and **mput**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in csh(1). For **mdelete** and **mget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file: the exact result depends on the foreign operating system and ftp server, and can be previewed by doing '**mls** *remote-files* -'. Note: **mget** and **mput** are not meant to transfer entire directory subtrees of files. That can be done by transferring a tar(1) archive of the subtree (in binary mode).

hash Toggle hash-sign ("#") printing for each data block transferred. The size of a data block is 1024 bytes.

help [*command*]
Print an informative message about the meaning of *command*. If no argument is given, *ftp* prints a list of the known commands.

lcd [*directory*]
Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

ls [*remote-directory*] [*local-file*]
Print an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If no local file is specified, or if *local-file* is -, the output is sent to the terminal.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets '$' and '\' as special characters. A '$' followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A '$' followed by an 'i' signals that macro processor that the executing macro is to be looped. On the first pass '$i' is replaced by

the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A '\' followed by any character is replaced by that character. Use the '\' to prevent special treatment of the '$'.

mdelete [*remote-files*]
Delete the *remote-files* on the remote machine.

mdir *remote-files local-file*
Like **dir**, except multiple remote files may be specified. If interactive prompting is on, *ftp* will prompt the user to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget *remote-files*
Expand the *remote-files* on the remote machine and do a **get** for each file name thus produced. See **glob** for details on the filename expansion. Resulting file names will then be processed according to *case*, *ntrans*, and *nmap* settings. Files are transferred into the local working directory, which can be changed with 'lcd directory'; new local directories can be created with '! mkdir directory'.

mkdir *directory-name*
Make a directory on the remote machine.

mls *remote-files local-file*
Like **ls**, except multiple remote files may be specified. If interactive prompting is on, *ftp* will prompt the user to verify that the last argument is indeed the target local file for receiving **mls** output.

mode [*mode-name*]
Set the file transfer *mode* to *mode-name*. The default mode is "stream" mode.

mput *local-files*
Expand wild cards in the list of local files given as arguments and do a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting file names will then be processed according to *ntrans* and *nmap* settings.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. *Inpattern* is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences '$1', '$2', ..., '$9' in *inpattern*. Use '\' to prevent this special treatment of the '$' character. All other characters are treated literally, and are used to determine the **nmap** *inpattern* variable values. For exmaple, given *inpattern* $1.$2 and the remote file name "mydata.data", $1 would have the value "mydata", and $2 would have the value "data". The *outpattern* determines the resulting mapped filename. The sequences '$1', '$2', ..., '$9' are replaced by any value resulting from the *inpattern* template. The sequence '$0' is replace by the original filename. Additionally, the sequence '[*seq1*,*seq2*]' is replaced by *seq1* if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command "nmap $1.$2.$3 [$1,$2].[$2,file]" would yield the output filename "myfile.data" for input filenames "myfile.data" and "myfile.data.old", "myfile.file" for the input filename "myfile", and "myfile.myfile" for the input filename ".myfile". Spaces may be included in *outpattern*, as in the example: nmap $1 sed "s/ *$//" > $1 . Use the '\' character to prevent special treatment of the '$', '[', ']', and ',' characters.

ntrans [*inchars* [*outchars*]]
Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

open *host* [*port*]
Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, *ftp* will attempt to contact an FTP server at that port. If the *auto-login* option is on (default), *ftp* will also attempt to automatically log the user in to the FTP server (see below).

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** will transfer all files, and any **mdelete** will delete all files.

proxy *ftp-command*
Execute an ftp command on a secondary control connection. This command allows simultaneous connection to two remote ftp servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command "proxy ?" to see other ftp commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** will not define new macros during the auto-login process, **close** will not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the ftp protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any *ntrans* or *nmap* settings in naming the remote file. File transfer uses the current settings for *type*, *format*, *mode*, and *structure*.

pwd Print the name of the current working directory on the remote machine.

quit A synonym for **bye**.

quote *arg1 arg2 ...*
The arguments specified are sent, verbatim, to the remote FTP server.

recv *remote-file* [*local-file*]
A synonym for get.

remotehelp [*command-name*]
Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

rename [*from*] [*to*]
Rename the file *from* on the remote machine, to the file *to*.

reset Clear reply queue. This command re-synchronizes command/reply sequencing with the remote ftp server. Resynchronization may be neccesary following a violation of the ftp protocol by

the remote server.

rmdir *directory-name*
Delete a directory on the remote machine.

runique
Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a get or mget command, a ".1" is appended to the name. If the resulting name matches another existing file, a ".2" is appended to the original name. If this process continues up to ".99", an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that **runique** will not affect local files generated from a shell command (see below). The default value is off.

send *local-file* [*remote-file*]
A synonym for put.

sendport
Toggle the use of PORT commands. By default, *ftp* will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, *ftp* will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer. This is useful for certain FTP implementations which do ignore PORT commands but, incorrectly, indicate they've been accepted.

status Show the current status of *ftp*.

struct [*struct-name*]
Set the file transfer *structure* to *struct-name*. By default "stream" structure is used.

sunique
Toggle storing of files on remote machine under unique file names. Remote ftp server must support ftp protocol STOU command for successful completion. The remote server will report unique name. Default value is off.

tenex Set the file transfer type to that needed to talk to TENEX machines.

trace Toggle packet tracing.

type [*type-name*]
Set the file transfer *type* to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

user *user-name* [*password*] [*account*]
Identify yourself to the remote FTP server. If the password is not specified and the server requires it, *ftp* will prompt the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user will be prompted for it. If an account field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless *ftp* is invoked with "auto-login" disabled, this process is done automatically on initial connection to the FTP server.

verbose Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

? [*command*]
A synonym for help.

Command arguments which have embedded spaces may be quoted with quote (") marks.

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending a ftp protocol ABOR command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for ABOR processing. If the remote server does not support the ABOR command, an "ftp>" prompt will not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence will be ignored when *ftp* has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described above, or from unexpected behavior by the remote server, including violations of the ftp protocol. If the delay results from unexpected remote server behavior, the local *ftp* program must be killed by hand.

FILE NAMING CONVENTIONS

Files specified as arguments to *ftp* commands are processed according to the following rules.

1) If the file name "-" is specified, the **stdin** (for reading) or **stdout** (for writing) is used.

2) If the first character of the file name is "|", the remainder of the argument is interpreted as a shell command. *Ftp* then forks a shell, using *popen*(3) with the argument supplied, and reads (writes) from the stdout (stdin). If the shell command includes spaces, the argument must be quoted; e.g. ""|ls -lt"". A particularly useful example of this mechanism is: "dir |more".

3) Failing the above checks, if "globbing" is enabled, local file names are expanded according to the rules used in the *csh*(1); c.f. the *glob* command. If the *ftp* command expects a single local file (.e.g. **put**), only the first filename generated by the "globbing" operation is used.

4) For **mget** commands and **get** commands with unspecified local file names, the local filename is the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.

5) For **mput** commands and **put** commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters which may affect a file transfer. The *type* may be "ascii", "image" (binary), "ebcdic", or "local byte size". *Ftp* supports the ascii and image types of file transfer, plus local byte size 8 for **tenex** mode transfers.

Ftp supports only the default values for the remaining file transfer parameters: *mode*, *form*, and *struct*.

OPTIONS

Options may be specified at the command line, or to the command interpreter.

The **-v** (verbose on) option forces *ftp* to show all responses from the remote server, as well as report on data transfer statistics.

The **-n** option restrains *ftp* from attempting "auto-login" upon initial connection. If auto-login is enabled, *ftp* will check the *.netrc* (see below) file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, *ftp* will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.

The **-i** option turns off interactive prompting during multiple file transfers.

The **-d** option enables debugging.

The **-g** option disables file name globbing.

THE .netrc FILE

The .netrc file contains login and initialization information used by the auto-login process. It resides in the user's home directory. The following tokens are recognized; they may be separated by spaces, tabs, or new-lines:

machine *name*
: Identify a remote machine name. The auto-login process searches the .netrc file for a **machine** token that matches the remote machine specified on the *ftp* command line or as an **open** command argument. Once a match is made, the subsequent .netrc tokens are processed, stopping when the end of file is reached or another **machine** token is encountered.

login *name*
: Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.

password *string*
: Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the .netrc file, *ftp* will abort the auto-login process if the .netrc is readable by anyone besides the user.

account *string*
: Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

macdef *name*
: Define a macro. This token functions like the *ftp* **macdef** command functions. A macro is defined with the specified name; its contents begin with the next .netrc line and continue until a null line (consecutive new-line characters) is encountered. If a macro named *init* is defined, it is automatically executed as the last step in the auto-login process.

BUGS

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD UNIX ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

© 1989 NeXT, Inc. All Rights Reserved.
NeXT, the NeXT logo, Workspace Manager, Application Kit, Interface Builder, Digital Librarian, Sound Kit, Music Kit and Jot are trademarks of NeXT, Inc. NeXTStep is a registered trademark of NeXT, Inc. UNIX is a registered trademark of AT&T. Ethernet is a registered trademark of Xerox Corporation. PostScript and Display PostScript are trademarks of Adobe Systems Incorporated. Sun Microsystems, Inc. is the developer and owner of the Network File System. WriteNow is a trademark licensed to NeXT, Inc.